Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Detekcia bleskov v rádiových spektrogramoch založená na konvolučných neurónových sieťach

Bakalárska práca

2021 **Adrián Kundrát**

Technická univerzita v Košiciach

Fakulta elektrotechniky a informatiky

Detekcia bleskov v rádiových spektrogramoch založená na konvolučných neurónových sieťach

Bakalárska práca

Študijný program:	Hospodárska informatika
Študijný odbor:	Informatika
Školiace pracovisko:	Katedra kybernetiky a umelej inteligencie (KKUI)
Školiteľ:	doc. Ing. Peter Butka, PhD.
Konzultant:	Ing. Viera Maslej Krešňáková
	RNDr. Šimon Mackovjak, PhD.

Košice 2021 **Adrián Kundrát**

Abstrakt v SJ

Súčasné meracie aparatúry dokážu zaznamenať úder blesku na vzdialenosť tisícov kilometrov. Je to vďaka meraniam elektromagnetických pulzov v rádiovej oblasti do 30 kHz (VLF). Následnou analýzou je možné nielen lokalizovať úder blesku, ale aj charakterizovať vlastnosti prostredia medzi bleskom a detektorom. Techniky strojového učenia môžu výrazne zjednodušiť detekciu a analýzu týchto pulzov zaznamenaných v tzv. spektrogramoch. Cieľom bakalárskej práce bude oboznámiť sa so základmi konvolučných neurónových sietí (CNN) a s ich využitím na detekciu špecifických udalostí (elektromagnetických pulzov) na obrázkoch (spektrogramoch).

Kľúčové slová

hlboké učenie, neurónová sieť, detekcia objektov, spektrogram, blesk, disperzia, tweek

Abstrakt v AJ

Current measuring devices could detect lightning strokes thousands of kilometers away. It results from measurements of electromagnetic pulses in a very low frequency (VLF) range below 30 kHz. By subsequent analysis, it is possible to locate lightning stroke and characterize properties of the environment between lightning stroke and detector. Machine learning techniques can significantly simplify the detection and analysis of those pulses recorded in spectrograms. The goal of this bachelor thesis is to get acquainted with the basics of convolutional neural networks (CNN) and with their usage in the detection of specific events (electromagnetic pulses) on figures (spectrograms).

Kľúčové slová v AJ

deep learning, neural network, object detection, spectrogram, lightning stroke, dispersion, tweek

61925

TECHNICKÁ UNIVERZITA V KOŠICIACH

FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Katedra kybernetiky a umelej inteligencie

ZADANIE BAKALÁRSKEJ PRÁCE

Študijný odbor: **Informatika**

Študijný program: **Hospodárska informatika**

Názov práce:

Detekcia bleskov v rádiových spektrogramoch založená na konvolučných neurónových sieťach

Lightning detection in radio spectrograms based on convolutional neural networks

Študent: **Adrián Kundrát**

Školiteľ: **doc. Ing. Peter Butka, PhD.**

Školiace pracovisko: **Katedra kybernetiky a umelej inteligencie**

Konzultant práce: **Ing. Viera Maslej Krešňáková, RNDr. Šimon Mackovjak, PhD.**

Pracovisko konzultanta: **Ústav experimentálnej fyziky SAV**

Pokyny na vypracovanie bakalárskej práce:

1. Podať teoretický prehľad problematiky hlbokého učenia a jeho využitia pre analýzu rádiových spektrogramov.
2. Analyzovať a vhodne predspracovať dostupné dáta, navrhnúť postup detekcie bleskov v rádiových spektrogramoch.
3. Implementovať a otestovať navrhnutý postup na predspracovaných dátach.
4. Vypracovať dokumentáciu podľa pokynov katedry a vedúceho práce.

Jazyk, v ktorom sa práca vypracuje: slovenský

Termín pre odovzdanie práce: 28.05.2021

Dátum zadania bakalárskej práce: 30.10.2020

.....

prof. Ing. Liberios Vokorokos, PhD.

dekan fakulty

Čestné vyhlásenie

Vyhlasujem, že som bakalársku prácu vypracoval samostatne s použitím uvedenej odbornej literatúry.

Košice 28. 5. 2021 .....

Vlastnoručný podpis

Poďakovanie

Chcel by som sa poďakovať môjmu školiteľovi, doc. Ing. Petrovi Butkovi, PhD. za cenné rady a pripomienky k práci. Taktiež veľká vďaka patrí konzultantovi práce, RNDr. Šimonovi Mackovjakovi, PhD. za ochotu a rady nie len v oblasti domény. Táto práca vznikla najmä vďaka Ing. Ivane Kolmašovej, PhD. a prof. RNDr. Ondřejovi Santolíkovi, Dr., ktorým chcem rovnako poďakovať. V neposlednom rade moja vďaka patrí konzultantke Ing. Viere Maslej Krešňákovej za nespočetné množstvo času, nesmierne veľa ochoty a trpezlivosti pri konzultáciách a určite aj za nekonečnú energiu a motiváciu za čo najlepšími výsledkami. Veľká vďaka patrí aj mojej rodine za morálnu podporu, ktorá mi bola prejavovaná v každej možnej situácii.

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam symbolov a skratiek

AdaGrad Adaptive Gradient Algorithm

Adam Algoritmus adaptívnej optimalizácie momenta

BCE Binary Cross Entropy

CCSW Cross-Correlation with a Simulated Whistler

CE Cross Entropy

CNN Convolutional Neural Network

CSP Cross Stage Partial

ELU Exponential Linear Unit

FN False Negative

FP False Positive

IoU Intersection over Union

KPI Key performance indicator

PANet Path Aggregation Network for Instance Segmentation

ReLU Recified Linear Unit

ResNet Residual Networks

RGB Red Green Blue color canal

RMSProp Root Mean Square Propagation

SDCNN Sliding Deep Convolutional Neural Network

SGD Stochastic Gradient Descent

SSD Single-Shot Detector

TN True Negative

TP True Positive

VLF Very low frequency

YOLO You Only Look Once

Úvod

Neurónové siete a téma detekcie objektov (angl. *object detection*) sú v súčasnosti bežným ľuďom bližšie, ako by sa mohlo zdať. Samotná technológia detekcie objektov pomocou neurónových sietí v posledných rokoch zažíva nevídaný rozmach. Túto technológiu je vďaka jej rýchlosti, efektívnosti a presnosti možné použiť takmer všade, kde je potrebné na istom obrazovom zdroji niečo nájsť. Či už v témach medicíny, napríklad pri vyhľadávaní rôznych oblastí na pľúcach napadnutých rakovinou (Jaeger et al., 2020), tak aj v bežnom živote, ako napríklad čoraz viac aktuálne technológie autonómneho riadenia v autách. Ako príklad takejto technológie môžeme uviezť spoločnosť Tesla[1] a jej technológiu autonómneho riadenia *AutopilotAI*, ktorej jedna z používaných techník je práve detekcia objektov. V tejto práci bude detekcia objektov využívaná na analýzu spektrogramov ako výstupov z meracích prístrojov.

Veda sa pohla za posledné desaťročia míľovými krokmi vpred a preto sa v súčasnosti už neriešia iba problémy nedostatku dát, ako tomu bolo niekedy. Do popredia sa dostáva pojem *Big Data* a nové výzvy založené na spracovaní týchto dát. Nedostatok dátových analytikov vo väčšine odborov v dnešnej dobe spôsobuje to, že existuje množstvo doposiaľ neanalyzovaných a potenciálne užitočných dát.

Spolupráca doménových expertov s informatikmi by ale mohla priniesť efektívne získané výsledky. Práve vďaka takejto spolupráci mohla vzniknúť aj táto práca. Ide o spoluprácu s Ústavom fyziky atmosféry Akadémie vied Českej republiky a Ústavom experimentálnej fyziky Slovenskej akadémie vied. Konkrétne sa bude jednať o spracovanie veľkého množstva dát zachytených v horizonte niekoľkých rokov zo zariadenia ELMAVAN-G, ktoré je vyvíjané Akadémiou vied ČR.

V prvej kapitole je popísaný teoretický úvod do neurónových sietí, detekcie objektov a použitých technológií (YOLO). V tejto kapitole sa čitateľ oboznámi so základnými princípmi fungovania týchto techník, priebehom ich učenia a regularizácie.

[1] `https://www.tesla.com/autopilotAI`

Ďalej sa dozvie základné informácie o crowdsourcingu a platforme *Zooniverse*, pomocou ktorej bol vytvorený anotačný projekt.

Druhá kapitola popisuje analýzu súčasného stavu, kde sú porovnané štyri existujúce práce zaoberajúce sa podobnou doménou.

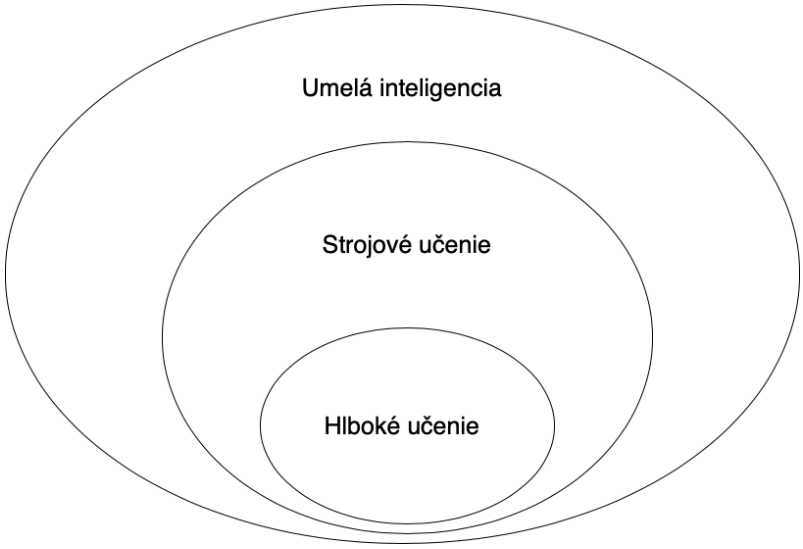
V úvode tretej kapitoly je popísaný zber a nadobudnutie dát použitých v práci. Táto časť ponúka informácie o analyzátore, ktorý slúžil ako zdroj dát. Nasleduje prierez celou technickou realizáciou danej práce. Začína sa pochopením cieľa a dát, kde sú podrobne popísané detaily o požadovanom výstupe a o poskytnutých vstupoch, s ktorými sme pracovali. Pokračuje popisom prípravy dát a vytvorením anotačného projektu na tvorbu datasetu. Dataset pozostával z obrázkov ako vstupnej množiny určenej na trénovanie neurónovej siete. Nasleduje modelovanie s popisom využitých parametrov pri trénovaní a optimalizácii. V ďalšej podkapitole je popísané vyhodnotenie trénovania spolu s metrikami. Následne je popísaný *post-processing*, na základe ktorého boli vytvorené ďalšie výstupy požadované v úvode. Prácu uzatvára nasadenie, kde je stručne popísaný ďalší plánovaný postup používania programu.

1 Neurónové siete a detekcia objektov

Na aspoň čiastočné pochopenie toho, o čo sa v problematike hlbokého učenia jedná, je potrebné v prvom rade pracovať s faktom, že nejde o štandardnú programovaciu paradigmu. Princíp hlbokého učenia nie je založený na princípe programátora tvoriaceho program, ktorý bude schopný vyriešiť na základe daných pravidiel istý vstupný problém. Hlboké učenie pracuje s myšlienkou, či je program schopný sa naučiť klasifikovať alebo predikovať nové prípady na základe automatickej extrakcie potrebných príznakov. Slovíčko učenie je v tomto kontexte podstatné, keďže sa program učí na základe princípov rozlišovania unikátnych znakov, príznakov a štatisticky vyjadrovaných vzťahov medzi vstupmi. Vďaka týmto naučeným aspektom je potom program schopný vytvoriť tzv. pravidlá a podmienky, vďaka ktorým vie automaticky vyhodnocovať vstupy. Základná otázka pri hlbokom učení teda môže byť skomponovaná nasledovne: Je program schopný vytvoriť a naučiť sa pravidlá automaticky namiesto toho, aby mu tie pravidlá museli byť dané programátorom (Chollet et al., 2018)? A práve táto otázka poukazuje na základný rozdiel medzi strojovým a hlbokým učením. Zatiaľ čo pri strojovom učení musí potrebné príznaky extrahovať človek, hlboké učenie dokáže tieto príznaky extrahovať automaticky.

Na Obrázku 1 – 1 je možné vidieť Vennov diagram, na základe ktorého môžeme povedať, že strojové učenie je podmnožinou umelej inteligencie a hlboké učenie podmnožinou strojového učenia. Pojem hlboké učenie pre neurónové siete poukazuje na hĺbku modelu, ktorý sa na dátach učí. Táto hĺbka vyjadruje počet vrstiev, ktoré daná neurónová sieť obsahuje. V súčasnosti sa neurónové siete skladajú rádovo z niekoľkých desiatok až stovák vrstiev. Tieto vrstvy učením vyhodnocujú vstupné dáta a vytvárajú si štatistické hodnoty pre jednotlivé triedy, ktoré budú neskôr pri klasifikáciách používať. Tieto hodnoty sa nazývajú váhy a počas učenia sa priebežne ukladajú a zdokonaľujú pre budúce použitie pri klasifikácii, či predikcii. V skratke je teda hlboké učenie pojem, ktorý charakterizuje proces analýzy vstupných dát a následného tvorenia váh pomocou neurónovej siete pre budúce použitie pri kla-

sifikácii (Chollet et al., 2018). V ďalších podkapitolách sa budeme venovať práve hlbokému učeniu a neurónovým sieťam, kde stručne popíšeme základné aktivačné, optimalizačné či regularizačné funkcie.



Obrázok 1 – 1 Porovnanie vzťahu medzi umelou inteligenciou, strojovým a hlbokým učením.

1.1 Úvod do neurónových sietí

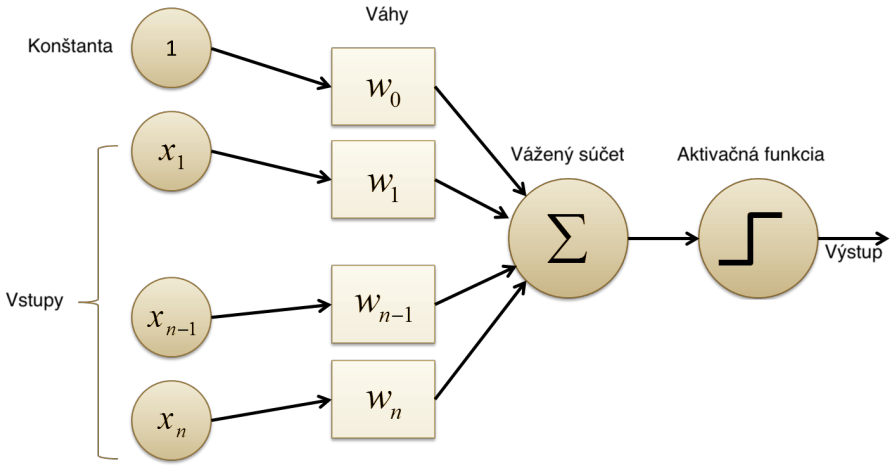
V úvodnej časti bolo popísané hlboké učenie. V tejto podkapitole sa pozrieme na to, čo vlastne neurónové siete sú a ako fungujú. Noriega (2005) približuje, že neurónové siete sú programovacia paradigma podobajúca sa na mikroštruktúru mozgu nielen zvonku, ale aj zvnútra. Táto informácia je založená na skutočnosti, že sa učia a zdokonaľujú na vlastných chybách a už nadobudnutých skúsenostiach. Môžeme teda skonštatovať, že neurónové siete fungujú podobne ako ľudský mozog. V oblasti umelej inteligencie sú neurónové siete používané na veľkom množstve úloh. Nasledujúci odsek bude zameraný na základné znaky perceptrónu, ktorý sa považuje za akúsi základnú neurónovú sieť.

1.1.1 Perceptrón

Perceptrón môžeme definovať ako binárne vyhodnocujúci algoritmus. Binárny z toho dôvodu, že sa používa na kontrolované učenie binárnych klasifikátorov. Vďaka tejto skutočnosti vieme perceptrón nazvať jednovrstvovou neurónovou sieťou. Perceptrón sa skladá z troch častí:

- Vstup (vstupné hodnoty),
- váhy a bias, súčet váh,
- aktivačná funkcia.

Perceptrón funguje na základe postupnosti naznačenej vo vyššie spomenutých častiach. Na začiatku pracuje so vstupnými hodnotami. Všetky tieto hodnoty vynásobí im prislúchajúcimi váhami. Po následnom sčítaní výstupných hodnôt násobenia vznikne vážený súčet. Vážený súčet je potom ďalej použitý ako vstup do aktivačnej funkcie, na ktorej výstupe je už výstup perceptrónu. Na základe tohto procesu dokážeme vyhodnotiť, že perceptrón je jedna vrstva neurónovej siete a slúži na klasifikáciu jednej triedy (Goodfellow et al., 2016). Základná vizualizácia perceptrónu je zobrazená na Obrázku 1 – 2.



Obrázok 1 – 2 Perceptrón.

1.1.2 Aktivačné funkcie

V tomto oddiele sa budeme venovať aktivačným funkciám. Aktivačné funkcie sú dôležitou súčasťou neurónových sietí. Používajú sa spravidla na transformáciu vstupného signálu na výstupný. V skutočnosti je daný výstupný signál napojený ako vstup do ďalšej vrstvy v poradí. V priebehu neurónovej siete sa najprv zrátajú vstupy spolu s váhami. Po následnej aplikácii aktivačnej funkcie vznikne výstup danej vrstvy. Tento výstup je použitý ako vstup do ďalšej vrstvy v skrytých vrstvách, alebo ako výstup z výstupnej vrstvy. Na výstupných vrstvách slúžia aktivačné funkcie ako klasifikátor pravdepodobnosti, že príklad patrí do danej triedy. Podľa toho môžeme aktivačné funkcie rozdeliť do skupín:

- **Aktivačné funkcie na skrytých vrstvách**:
  - Jednokroková funkcia (angl. *Binary step function*),
  - Lineárna funkcia,
  - Sigmoidálna funkcia – Sigmoid,
  - Hyperbolická tangens funkcia – Tanh,
  - ReLU a jej varianty *Leaky ReLU* a *Parametrized ReLU*,
  - *Exponential Linear Unit* – ELU,
  - Swish.
- **Aktivačné funkcie na výstupnej vrstve**:
  - Sigmoidálna funkcia – Sigmoid,
  - Normalizovaná exponenciálna funkcia – softmax.

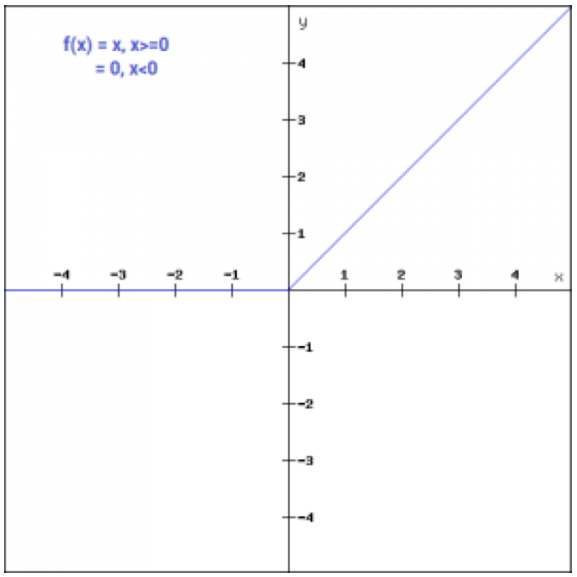
Ďalšia časť je zameraná na najpoužívanejšie aktivačné funkcie.

- **ReLU**

  ReLU je skratka pre *Rectified Linear Unit*. Jedná sa o často používanú nelineárnu aktivačnú funkciu. Jej hlavná funkcionalita je založená na skutočnosti, že sa neuróny neaktivujú po každom vstupe. Vďaka tejto skutočnosti je ReLU efektívnejšia ako iné aktivačné funkcie. Matematický predpis tejto funkcie je:

$$f(x) = \max(0, x). \tag{1.1}$$

  Aktivačná funkcia ReLU aktivuje neurón iba ak je hodnota na vstupe vyššia ako nula. Ináč ostáva neurón neaktívny, a teda ak je vstup akékoľvek záporné číslo alebo nula, výstup zodpovedá hodnote nula, ako je možné vidieť aj na Obrázku 1 – 3.



Obrázok 1 – 3 Priebeh aktivačnej funkcie ReLU (Sharma, 2017).

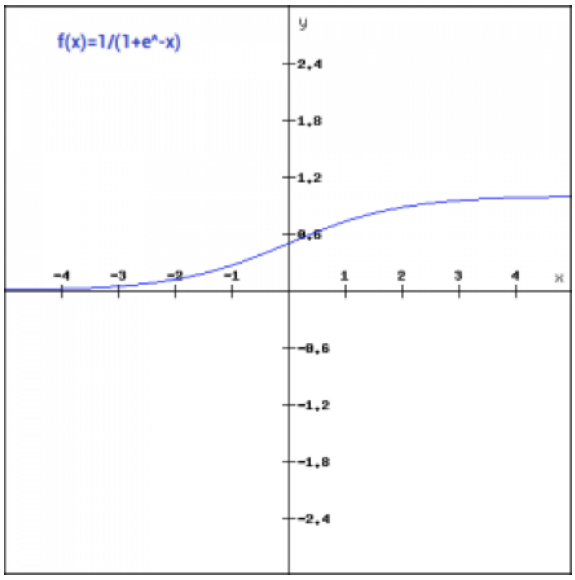
- **Sigmoid**

  Sigmoid je aktivačná funkcia, ktorá na vstupe požaduje reálnu hodnotu, z ktorej na výstupe vygeneruje hodnotu medzi 0 a 1. Táto aktivačná funkcia je veľmi obľúbená kvôli tomu, že sa s ňou ľahko pracuje a disponuje mnohými pozitívnymi vlastnosťami: je nelineárna a zároveň ani jej kombinácie nie sú lineárne,

diferencovateľná, monotónna a má nemenný rozsah výstupu. Z matematického hľadiska je táto aktivačná funkcia popísaná ako:

$$f(x) = \frac{1}{1 + e^{-x}}. \tag{1.2}$$

Priebeh aktivačnej funkcie Sigmoid je zobrazený na Obrázku 1 – 4. Spravidla sa táto funkcia používa na binárnu klasifikáciu.



Obrázok 1 – 4 Priebeh aktivačnej funkcie Sigmoid (Sharma, 2017).

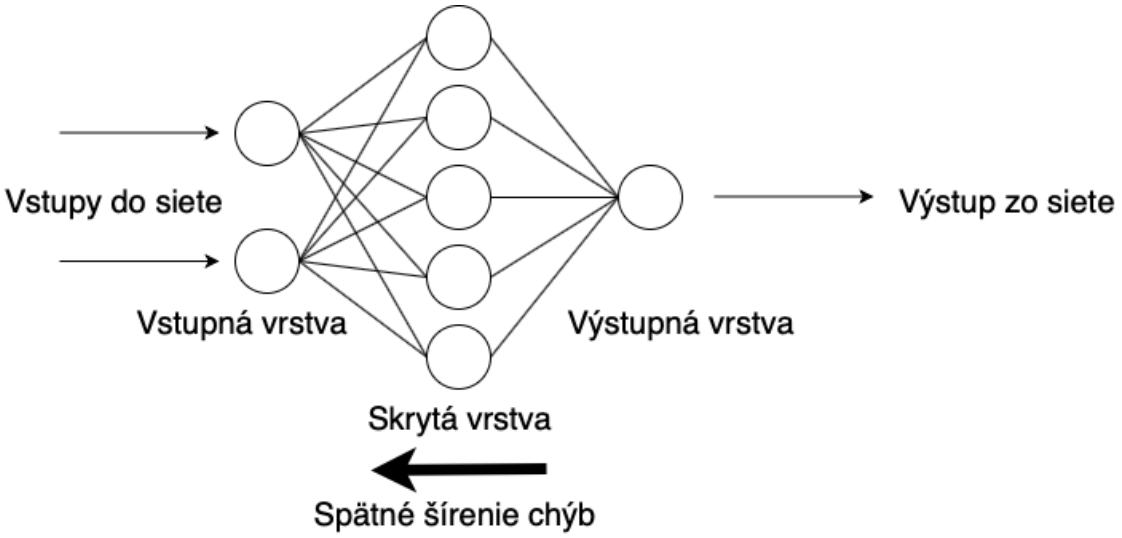
- **Softmax**

  Aktivačná funkcia softmax pozostáva z kombinácie viacerých funkcií typu Sigmoid. V porovnaní čisto s aktivačnou funkciou Sigmoid, ktorá sa používa na binárnu klasifikáciu, je aktivačná funkcia softmax používaná na riešenie problémov alebo prípadov, pri ktorých je potrebná klasifikácia do viacerých tried (Sharma, 2017; Brownlee, 2019). Matematické vyjadrenie tejto funkcie je dané vzťahom:

$$\sigma(z)_j = \frac{e^{z_j}}{\sum_{k=1}^{K} e^{z_k}}. \tag{1.3}$$

1.1.3 Chybové funkcie a spätné šírenie chýb

Spätné šírenie chýb (angl. *backpropagation*) je algoritmus na kontrolované učenie neurónových sietí. Metóda dokáže vyrátať gradient chybovej funkcie s ohľadom na váhy. V tomto algoritme prebieha prepočítavanie gradientu smerom späť po neurónovej sieti, kedy sa na základe chyby z výstupu siete spätne optimalizujú váhy. Tým pádom sa automaticky optimalizuje aj model a znižuje sa hodnota chybovej funkcie. Na prepočítavanie jednotlivých vrstiev sa používajú čiastkové výpočty tých predchádzajúcich (Chollet et al., 2018; McGonagle et al., 2018). Na Obrázku 1 – 5 je možné vidieť jednoduchý náhľad priebehu spätného šírenia chýb smerom od výstupu.



Obrázok 1 – 5 Náhľad priebehu spätného šírenia chýb.

Chybová funkcia (angl. *loss function*) je v neurónových sieťach používaná spravidla na číselné vyjadrenie chyby vzniknutej učením, ktorej hodnotu (angl. *loss score*) vieme dostať po každej epoche (časti učenia). Princíp funkcie je založený na sledovaní odchýlky aktuálneho výstupu trénovania od požadovaného. Na vypočítanie tejto hodnoty je použitá trénovacia množina a výstup z predikcie v rámci siete počas trénovania. Aj vďaka spätnému šíreniu chýb sa dokáže neurónová sieť zdokonaľovať a redukovať tak nesprávne predikcie (Chollet et al., 2018).

- **Binárna krížová entropia**

  Binárna krížová entropia (angl. *Binary cross-entropy*, skr. BCE) je chybová funkcia

$$\mathrm{BCE} = -\mathrm{y}\log(\hat{\mathrm{y}}) - (1-\mathrm{y})\log(1-\hat{\mathrm{y}}), \tag{1.4}$$

  kde premenná y predstavuje skutočnú hodnotu a premenná $\hat{y}$ predstavuje predikciu.

- **Krížová entropia**

  Krížová entropia (angl. *cross-entropy*, skr. CE) je chybová funkcia

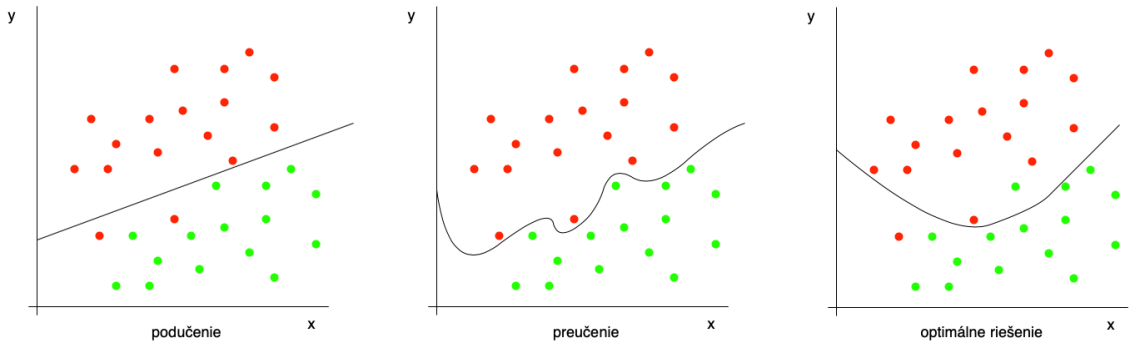
$$\mathrm{CE} = -\sum_{\mathrm{i}}^{\mathrm{M}} \mathrm{y_i}\log(\hat{\mathrm{y}}_\mathrm{i}), \tag{1.5}$$

  kde premenná y predstavuje skutočnú hodnotu, premenná $\hat{y}$ predstavuje predikciu a M je počet tried.

1.1.4 Priebeh učenia neurónových sietí

Táto kapitola je zameraná na chyby učenia modelu. Tieto chyby špecifikujú dva stavy, ktoré poukazujú na podučenie alebo preučenie modelu. Tieto stavy sú bežné pri neurónových sieťach a dá sa im predísť pomocou metód optimalizácie a regularizácie, ktoré sú objasnené v ďalších oddieloch.

Na prvej časti Obrázka 1 – 6 môžeme vidieť nedostatočne naučený, tzv. podučený model (angl. *underfitted*). Tento model nie je schopný extrahovať potrebné príznaky s požadovanou presnosťou. Vyznačuje sa vysokou mierou chyby ako na trénovacej, tak aj na testovacej množine. Na základ správneho vývoja chyby modelu na testovacej a validačnej množine vieme buď pridať viac epoch, alebo zvýšiť zložitosť modelu. Niekedy je však dataset natoľko špecifický, že ani tieto kroky nepomôžu. V tom prípade by bolo vhodné rozšíriť dátovú množinu napríklad augumentačnými technikami, vďaka čomu bude model mať väčšiu šancu naučiť sa potrebné príznaky.



Obrázok 1 – 6 Tri situácie výsledku učenia modelu.

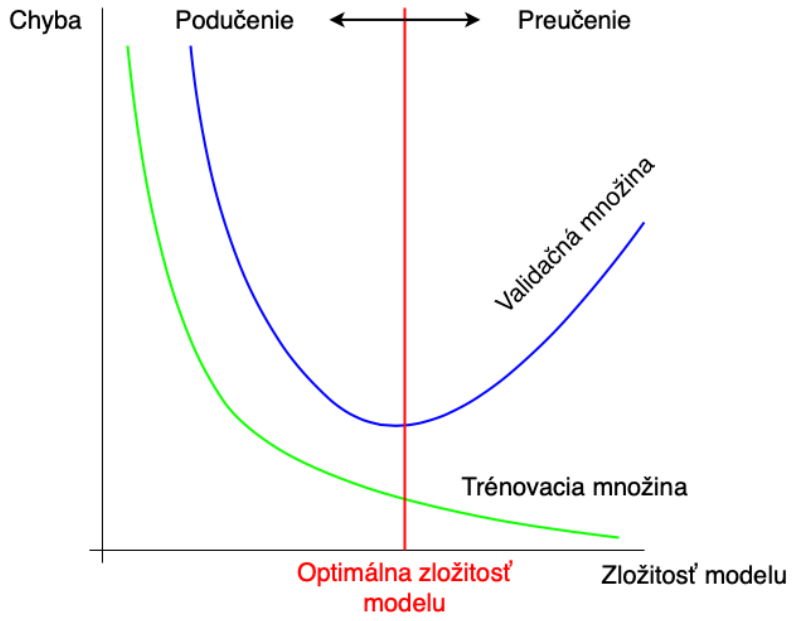
Na druhej časti Obrázka 1 – 6 vidíme takzvaný preučený model (angl. *overfitted*). O tomto modeli môžeme povedať, že model nie je dobre naučený vyhodnocovať, pretože stratil schopnosť generalizovať vlastnosti potrebné na klasifikáciu nových prípadov. Základný problém pri preučení spočíva v tom, že sa model naučil vyhodnocovať príliš špecifické vlastnosti dát na trénovacej množine, čo je pre testovaciu množinu kontraproduktívne. Model totiž v tomto prípade očakáva na testovacej časti takmer totožný vstup ako množina, na ktorej sa naučil. Výsledok toho je, že model nevie vyhodnocovať prípady, ktoré sa akokoľvek líšia od trénovacej množiny. Tým pádom vzniká jav, kedy je chyba na trénovacej množine veľmi nízka a zároveň je chyba na validačnej a následne testovacej množine vysoká. Riešením tejto situácie je optimalizácia a regularizácia modelu.

Na tretej časti Obrázka 1 – 6 je vidieť požadovaný stav modelu, kedy je schopný optimálne vyhodnotiť udalosti aj na testovacej množine. V tomto prípade je vzťah medzi chybami oboch množín v správnom pomere a model je optimálne naučený.

Z tohto popisu môžeme vyhodnotiť, že základná podmienka na predchádzanie týmto chybám pri učení je hľadanie akéhosi stredu – optimálneho pomeru hodnôt chyby na testovacej a na validačnej množine, ako môžeme vidieť aj na Obrázku 1 – 7.

1.1.5 Optimalizácia učenia

Ako bolo spomenuté vyššie, vďaka spätnému šíreniu chýb dokážeme regulovať a optimalizovať učenie. Na základe použitého optimalizátora dokážeme upravovať váhy

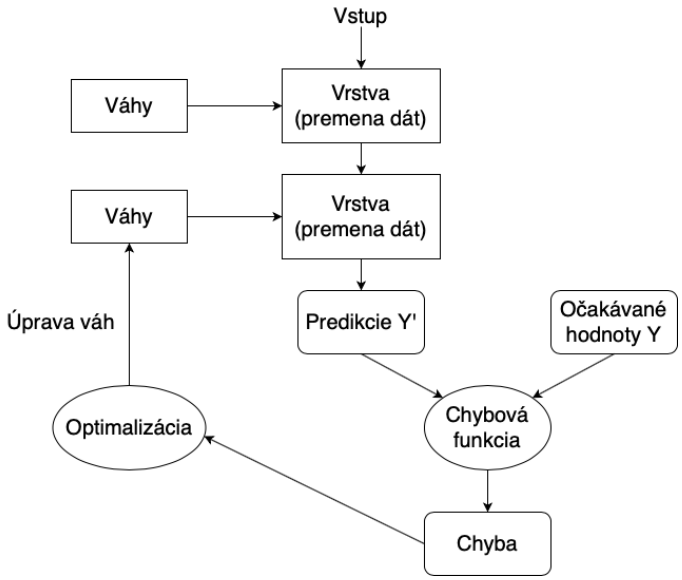


Obrázok 1 – 7 Náhľad pomeru chyby trénovacej a validačnej množiny voči zložitosti modelu.

v procese učenia tak, aby sme minimalizovali chybu učenia. Zároveň vďaka optimalizácii vieme regulovať rýchlosť učenia. Obrázok 1 – 8 znázorňuje princíp funkcie spätného šírenia chýb a optimalizácie váh. Optimalizačnou metódou, ktorá sa stala základom pre ďalšie optimalizátory, je SGD. Veľmi efektívny je však aj optimalizátor Adam, ktorý si popíšeme nižšie.

Stochastic gradient descent

Stochastická metóda gradientového zostupu (angl. *Stochastic gradient descent* skr. SGD, Bottou (2012)) je optimalizačná metóda, ktorá funguje na princípe aktualizácie váh po využití každého bodu, čiže každej časti datasetu. Vďaka tomu sa model optimalizuje pravidelne počas trénovania. To znamená, že táto metóda je iteratívna a v každej iterácii mení parametre tak, aby sa postupne priblížili k minimálnej hodnote funkcie. Pri tejto metóde gradientového zostupu využívame vlastnosti gradientu, pričom gradient funkcie v bode určuje smer najväčšieho rastu funkcie. Keďže chceme túto funkciu minimalizovať, posúvame váhy v opačnom smere.



Obrázok 1 – 8 Spätné šírenie chýb s úpravou váh optimalizáciou.

Optimalizátor Adam

Adaptive moment estimation (skr. Adam, Kingma and Ba (2014)) je veľmi efektívna metóda pri práci s veľkými modelmi obsahujúcimi mnoho parametrov. Je to optimalizačný algoritmus a je spojením výhod dvoch ďalších populárnych metód:

- *Adaptive Gradient Algorithm* (skr. AdaGrad, Duchi et al. (2011)),
- *Root Mean Square Propagation* (skr. RMSProp, Tieleman and Hinton (2012)).

To znamená, že Adam kombinuje výhody momentovej metódy (RMSProp) a adaptívnej zmeny rýchlosti učenia (AdaGrad) nezávislej pre každý parameter.

1.1.6 Regularizácia učenia

Regularizácia je pojem pre systém stratégií používaný v strojovom učení, ktorý sa snaží o znižovanie chyby pri testovaní aj napriek zvyšovaniu trénovacej chyby. Takmer všetky tieto stratégie sú založené na regulovaní učenia, ktorého základ je zvyšovanie bias, čoho následok je redukcia odchýlky pri detekcii. Dobrá regularizačná stratégia je založená na optimálnom pomere biasu a chyby pri detekcii, kedy sa

očakáva nízke zvyšovanie bias pri dostatočnom znížení chyby detekcie (Goodfellow et al., 2016). V konečnom dôsledku teda regularizácia slúži primárne na regulovanie vstupov do jednotlivých vrstiev neurónovej siete, aby nedochádzalo k preučeniu modelu.

Dropout

Dropout je jedna z najpoužívanejších a najefektívnejších regularizačných techník. Princíp fungovania tejto techniky je založený na náhodnom nastavovaní časti výstupných hodnôt vrstvy na nulu. Na vstupe tejto funkcie sa nastavuje hodnota *dropout rate*, ktorá reprezentuje podiel hodnôt z množiny výstupov, ktorý bude nastavený na 0 (Chollet et al., 2018).

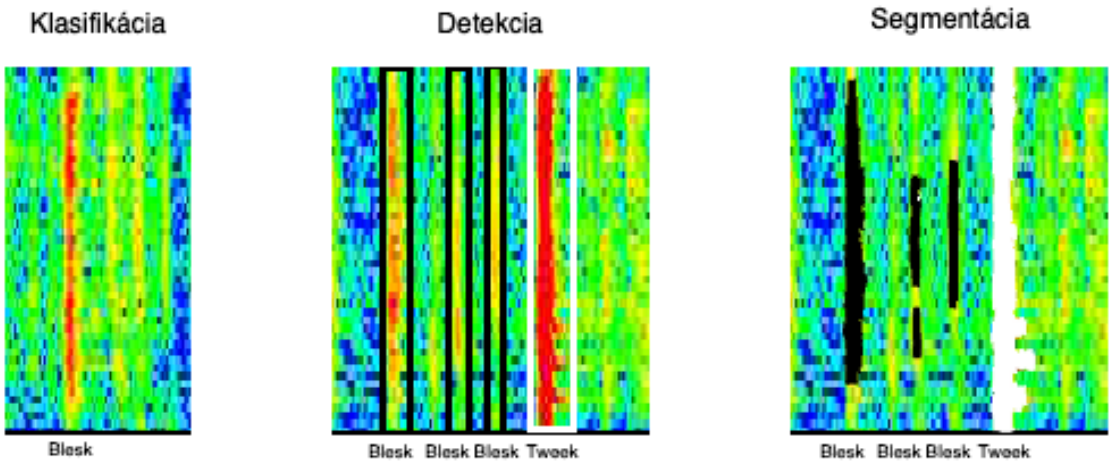
1.2 Spracovanie obrazu pomocou neurónových sietí

Klasifikácia obrázkov

Klasifikácia obrázkov (angl. *image classification*) je technika na rozpoznávanie obsahu obrázka. Ako nám názov môže napovedať, ide iba o tzv. zatriedenie obrázka, resp. priradenie triedy obrázku ako celku. Klasifikácia nedokáže určiť polohu objektu na obrázku. Tým pádom nedokáže ani vytvoriť tzv. *bounding box* (ohraničujúci rám) alebo masku na mieste, kde sa nachádza daný objekt, ako to dokáže napríklad detekcia objektov, resp. segmentácia obrázkov. Klasifikácia obrázkov je teda jednoduchá technika slúžiaca na vyhodnotenie objektov na obrázku a priradenie jednej výstupnej triedy, ktorá objektom patrí. Prvá časť Obrázka 1 – 9 znázorňuje princíp klasifikácie obrazu.

Detekcia objektov

Detekcia objektov (angl. *object detection*) je technika počítačového videnia, ktorá nám umožňuje pracovať s obrazom. Vďaka tejto technike je možné detegovať a lokalizovať objekty na obrázkoch, no aj v reálnom čase na obraze kamery alebo vo videách.



Obrázok 1 – 9 Náhľad porovnania klasifikácie, detekcie objektov a segmentácie priamo na obrázkoch používaných v tejto práci.

Technika detekcie objektov nám sprostredkúva nielen možnosť zistiť, aký konkrétny objekt sa na obrázku nachádza, ale skrze túto techniku vieme dostať aj relatívne presné súradnice polohy daného objektu, resp. jeho ohraničujúceho rámu. Spravidla na výstupe detekčnej neurónovej siete dostaneme súradnice daného rámu a jeho obsah v podobe triedy, ktorá vyjadruje typ objektu (Redmon et al., 2016). Náhľad funkcie tejto techniky je zobrazený v druhej časti Obrázka 1 – 9.

Segmentácia obrázkov

Segmentácia obrázkov (angl. *image segmentation*) je ďalšia z techník hľadania objektov na obrázku. Na rozdiel od detekcie objektov je výstup segmentácie presnejší. Zatiaľ čo pri detekcii je výstup modelu obdĺžnik okolo lokalizovaného objektu, výstupom zo segmentácie je maska. Maska zastupuje vrstvu ohraničujúcu objekt s presnosťou na pixely podľa jeho hrán. Segmentáciu môžeme definovať ako pridelenie triedy pre každý pixel obrázka. Na základe toho sú ďalej spojené pixely s rovnakou triedou s ohľadom na ich vizuálne vlastnosti. Vďaka tomu dostávame na výstupe nielen nájdený objekt, ale takisto aj jeho tvar (Ghosh et al., 2019). Náhľad funkcie

tejto techniky je zobrazený v tretej časti Obrázka 1 – 9.

V predchádzajúcich troch odsekoch sú priblížené základné techniky kontrolovaného učenia na prácu s obrazom a jeho obsahom. V našej práci sme sa venovali práve detekcii objektov. Táto technika sa nám javila ako najlepšia či už z pohľadu anotácie pre trénovanie (označovanie objektov na trénovacom datasete), tak aj z pohľadu samotného výstupu predikcií. Detekcia objektov v našej práci bola vykonaná neurónovou sieťou YOLO, ktorá bude popísaná v nasledujúcej časti.

1.2.1 YOLO

You Only Look Once (skr. YOLO, Redmon et al. (2016)) je unifikovaný algoritmus založený na princípe konvolučných neurónových sietí. Tento algoritmus je určený na detekciu objektov v reálnom čase. YOLO je založené na architektúre jednej neurónovej siete na predikciu ohraničujúcich rámčekov a klasifikáciu pravdepodobností priamo z celých obrázkov. Od uverejnenia prvej verzie YOLO neurónovej siete v roku 2016 tento algoritmus prešiel mnohými úpravami a vylepšeniami. Odvtedy si našiel veľa fanúšikov pre svoju jednoduchosť, rýchlosť a presnosť. Najaktuálnejšia verzia bola v čase písania tejto bakalárskej práce YOLOv5 Jocher et al. (2021), ktorá bola zároveň použitá v tejto práci na detekciu objektov (Yap et al., 2020).

Ako sme spomínali vyššie, základom YOLO algoritmu je konvolučná neurónová sieť (angl. *convolutional neural network*, skr. CNN). CNN je algoritmus, ktorý je schopný na vstupe (napr. obrázok) priradiť jednotlivým objektom na obrázku váhu a odlíšiť ich navzájom. V porovnaní s inými klasifikačnými algoritmami požadujú CNN oveľa menej predprípravy dát. Princíp konvolučných neurónových sietí je podobný princípu neurónov v ľudskom mozgu a bol inšpirovaný tzv. vizuálnou kôrou mozgu (zadná časť mozgu, angl. *visual cortex*), ktorá je zodpovedná za spracovanie stimulov z očí. Konvolučná neurónová sieť dokáže úspešne podchytiť časové a priestorové závislosti na obrázkoch vďaka naučeným filtrom. Vďaka nim dokáže sieť redukovať počet použitých parametrov pri detekcii. CNN sa skladá troch základných vrstiev. Jedna sa o konvolučnú, vzorkovaciu a plne pripojenú vrstvu. Konvolučná

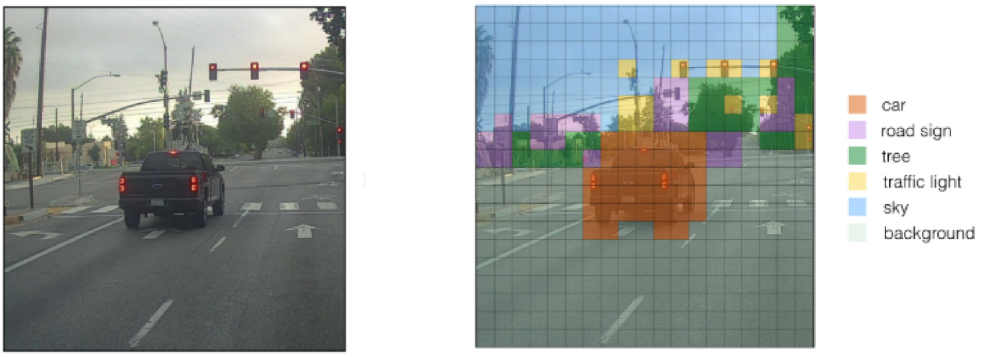
vrstva sa stará sa o extrakciu základných odlišujúcich vlastností obrázka, ako sú napríklad hrany či prechody v prvých vrstvách a v každej ďalšej vrstve sú to komplexnejšie tvary charakterizujúce hľadané objekty. Vzorkovacia vrstva sa v skratke stará o znižovanie priestorovej veľkosti obrázka kvôli znižovaniu požiadaviek na výpočtový výkon zariadenia. Plne pripojené vrstvy predstavujú doprednú neurónovú sieť, ktorá slúži na učenie sa nelineárnych kombinácií funkcií, pričom posledná plne pripojená vrstva predstavuje výstup konvolučnej neurónovej siete (Saha, 2018).

Na základe skutočnosti, že YOLO dokáže predikovať triedy a ohraničujúce boxy pre celý obrázok jediným behom algoritmu, sa táto metóda stáva neporovnateľne rýchlejšou ako ostatné algoritmy určené na riešenie podobných problémov. Základom výstupu detekčného algoritmu je predikcia. Vo väčšine prípadov sa jedná o obdĺžniky, ktoré ohraničujú detegovaný objekt. Výstup každej detekcie algoritmu YOLO pozostáva zo 4 parametrov:

- Parametre stredu boxu,
- šírka a výška boxu,
- hodnota charakterizujúca triedu predikovaného objektu,
- pravdepodobnosť predikcie.

Rýchlosť predikcií algoritmu spočíva v tom, že je každý z obrázkov rozdelený do mriežky, zväčša o rozmere 19×19. Na základe tejto mriežky je každá jej bunka schopná predikovať určitý počet boxov. Ďalej na základe pravdepodobnosti algoritmus priradí najpravdepodobnejšiu triedu ku každej bunke. Na Obrázku 1 – 10 je zobrazený stav rozdelenia do mriežky s priradenými triedami pre jednotlivé bunky.

Následne, po počiatočnom rozdelení tried, je potrebné vykonať ďalší krok, tzv. *non-max suppression* (Neubeck and Van Gool, 2006), slúžiaci na redukciu boxov, ktoré sú veľmi blízko seba na základe metriky *Intersection over Union* (skr. IoU). Metrika IoU slúži na vyjadrenie pomeru prieniku a zjednotenia jednotlivých boxov.



Obrázok 1 – 10 Náhľad výsledku rozdelenia obrázka do mriežky a priradenia tried jednotlivým bunkám. Zdroj: `https://miro.medium.com/max/4800/1*ZkfC29ck9IqmZ7poyESVxQ.png`

V tomto kroku sú zmazané boxy, ktorých IoU je vyššie ako prahová hodnota. Ako výstup ostane iba najpresnejší box pre každý objekt (Redmon et al., 2016).

1.2.2 YOLOv5

YOLOv5[2] bola prvýkrát publikovaná v roku 2020 firmou Ultralytics LLC[3]. Keďže pôvodné verzie YOLO boli primárne vytvárané pre DarkNet framework, spoločnosť Ultralytics pracovala na vytvorení jeho implementácie pre framework PyTorch. Jedna z takýchto implementácií bola YOLOv3 od Ultralytics[4]. Pôvodne bola teda YOLOv5 vylepšená verzia YOLOv3 pre PyTorch. Medzičasom ale bola vydaná YOLOv4 pre DarkNet framework[5] a teda v záujme vyhnutia sa konfliktom s názvami spoločnosť Ultralytics zvolila názov YOLOv5 (Yap et al., 2020). YOLOv5 sa skladá z troch častí:

[2] `https://github.com/ultralytics/yolov5/`
[3] `https://ultralytics.com`
[4] `https://github.com/ultralytics/yolov3`
[5] `https://github.com/AlexeyAB/darknet`

- *Model Backbone*

  Prvá časť modelu je primárne určená na extrakciu vlastností z obrázka. Ako *Model Backbone* je pri YOLOv5 používaná sieť *Cross Stage Partial* (skr. CSP). Táto časť neurónovej siete beží pri trénovaní a je veľmi náročná na výpočtový výkon, keďže počas nej prebieha rozoznávanie špecifických vlastností objektov na každom trénovanom obrázku. CSP čiastočne rieši problém potreby vysokého výkonu zariadenia redukciou výpočtov o 20 %. Viac o tejto sieti je možné sa dočítať vo Wang et al. (2020).

- *Model Neck*

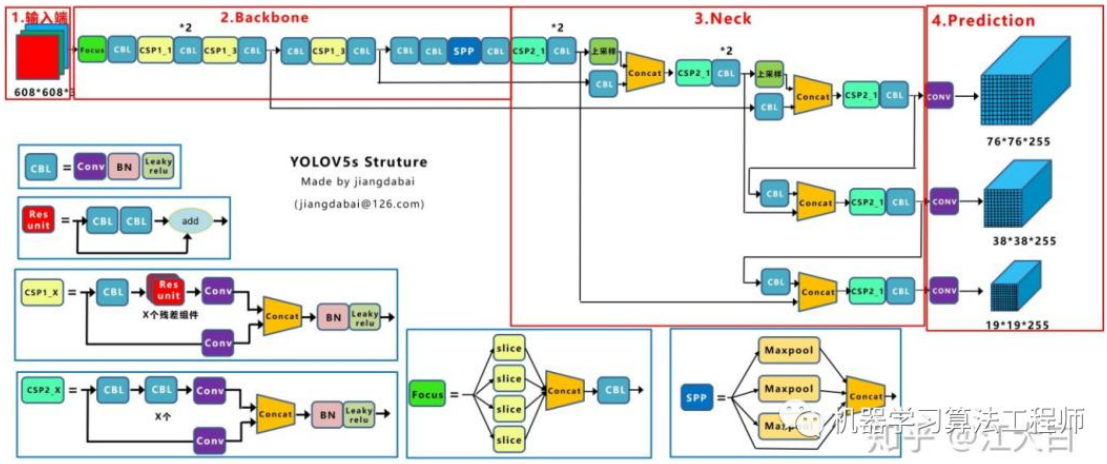
  Druhá časť modelu je používaná na tvorenie vlastnostných pyramíd. Tieto pyramídy pomáhajú modelu učiť sa vlastnosti objektov, čo v budúcnosti napomôže identifikácii rovnakého objektu, ktorý má odlišné vlastnosti. YOLOv5 používa na vytváranie vlastnostných pyramíd sieť *Path Aggregation Network for Instance Segmentation* (skr. PANet), ktorá je popísaná v článku Liu et al. (2018).

- *Model Head*

  Posledná časť modelu sa zameriava primárne na konečnú detekčnú časť modelu. Jej funkčnosť spočíva v generovaní finálnych rozhodnutí v klasifikácii, vytvorení hodnoty pravdepodobnosti a objekt ohraničujúcich obdĺžnikov na obrázku.

Na Obrázku 1 – 11 je možné vidieť architektúru neurónovej siete YOLO s rozdelením podľa častí spomenutých vyššie.

Čo sa aktivačných funkcií týka, YOLOv5 používa aktivačné funkcie Leaky ReLU a Sigmoid. Aktivačná funkcia Leaky ReLU je použitá na skrytých vrstvách, aktivačná funkcia Sigmoid je použitá na poslednej detekčnej vrstve. Ako základnú optimalizačnú funkciu používa YOLOv5 SGD. Zároveň je ale možné zvoliť ako optimalizačnú funkciu aj Adam. Z chybových funkcií YOLOv5 primárne používalo binárnu krížovú



Obrázok 1 – 11 Architektúra neurónovej siete YOLOv5. Zdroj: `https://tinyurl.com/ysy5t8bt`.

entropiu a *Focal loss* (Lin et al., 2017). Model, na ktorom sme trénovali a vykonávali detekciu bol zložený z 7,3 milióna parametrov a 224 vrstiev.

1.3 Crowdsourcing a Zooniverse

Táto kapitola je zameraná na bližší popis pojmu crowdsourcing. Podrobnejšie popíšeme tento prístup a spôsob, akým sme crowdsourcing využili počas našej práce. Ďalšia časť kapitoly sa zameriava na konkrétnu aplikáciu, ktorá bola pre účel tejto práce využitá a to, akým spôsobom na nej bol vytvorený anotačný projekt. Skrze tieto skutočnosti bolo možné vytvoriť dataset, na ktorom sme trénovali neurónovú sieť za účelom automatickej detekcie elektromagnetických pulzov.

Crowdsourcing je pojem, s ktorým sa čím ďalej tým viac stretávame aj v IT odvetví, kedy sú väčšinou potrebné výsledky za relatívne krátky čas. Keď sa zameriame na doslovný preklad názvu tohto prístupu, dokážeme v skratke popísať, čo daný prístup znamená. Pri preklade z anglického *crowd* – dav a *source* – zdroj prichádzame na to, že crowdsourcing je metóda používajúca ako zdroj informácií dav, v tomto ponímaní väčší počet ľudí alebo kolektív. V dnešnom svete, kedy si ako veľké korporácie, tak aj malé projekty uvedomujú, že tímová práca ľudí prináša efektivitu,

sa crowdsourcing teší veľkej obľube. Brabham (2008) definuje crowdsourcing ako delegovanie problému na väčšiu masu neznámych ľudí so zámerom jeho vyriešenia. V jednoduchosti je bežný priebeh crowdsourcingu nasledovný: organizácia zdieľa online problém, ktorý potrebuje riešiť. Vďaka tomuto kroku budú mať prístup ku problému jednotlivci, ktorí dokážu navrhnúť jeho riešenie. Tým pádom je organizácia schopná ťažiť z vonkajšieho zdroja a využiť to na svoj vlastný rast. Spravidla sú najlepšie riešenia ocenené istou formou kreditu a výsledkom tejto spolupráce je tzv. *win-win* situácia pre obe strany. Keďže je známe, že pri objemovo veľkých úlohách prináša vyššia kvantita riešiteľov aj rýchlejšie riešenie, sme crowdsourcing zvolili ako metódu pre anotáciu spektrogramov pre našu neurónovú sieť.

V súčasnosti existuje mnoho portálov, ktoré poskytujú možnosti sprostredkovania crowdsourcingu. Väčšina týchto portálov ale neposkytuje možnosť vytvorenia projektu na anotáciu dát, skôr sú stavané ako sprostredkovateľ riešenia z vlastných zdrojov na základe istej odmeny. My sme disponovali ľudskými zdrojmi a potrebovali sme iba vytvoriť anotačný projekt, na ktorom by mohol náš „dav“ anotovať. Vzhľadom na prechádzajúce skúsenosti na našej katedre sme po krátkom rešerši platforiem zvolili *Zooniverse*, ktorý sa javil byť pre naše potreby najlepší. Podľa *Zooniverse*[6] je táto platforma primárne určená pre profesionálov – či už vedcov, analytikov alebo výskumných pracovníkov. Platforma *Zooniverse* je orientovaná na pomoc výskumným pracovníkom, ktorí potrebujú efektívnu spoluprácu pri ich výskumnej činnosti. Hlavná výhoda tejto platformy je, že sa vďaka nej môže na výskumnej činnosti podieľať takmer každý. Nie je k tomu potrebné žiadne ďalšie vzdelanie alebo hardvérové príslušenstvo. Vďaka tejto jednoduchosti vie ktorýkoľvek dobrovoľník pristupovať k autentickým dátam, ktoré sú ďalej použité často aj na vedecký výskum. Takýto spôsob pomoci je výhodný pre obe strany, keďže takto jednoducho vie bežný človek prispieť ku vedeckým prácam a zároveň sa dostať ku zaujímavým dátam priamo zo zdroja.

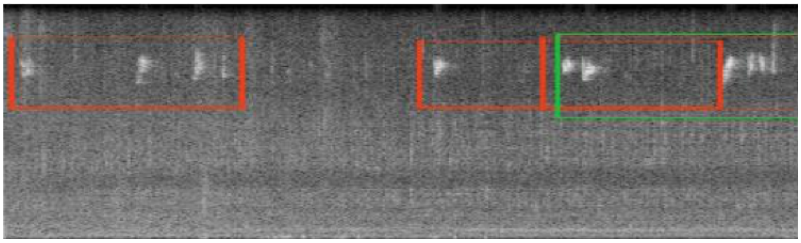
[6] `https://www.zooniverse.org/about`

2 Analýza súčasného stavu detekcie udalostí na spektrogramoch

V tejto kapitole bude vytvorená analýza súčasného stavu, kde bude popísaných niekoľko prác, ktoré spôsobom uskutočnenia alebo použitými technológiami súvisia s touto prácou.

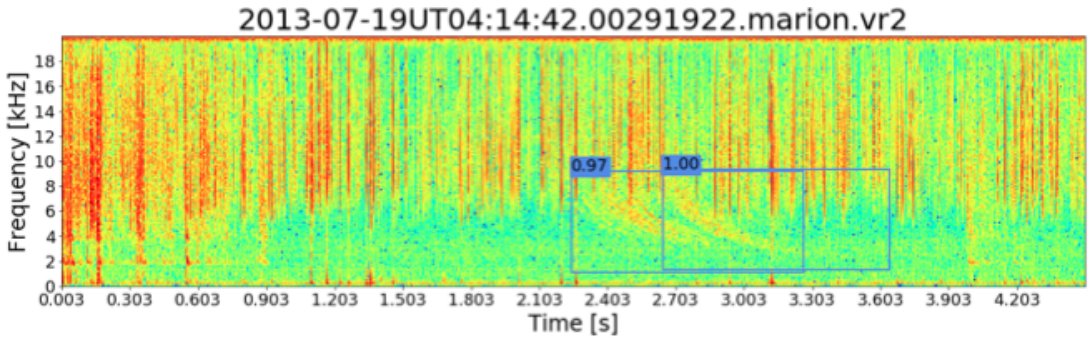
Práca Fanioudakis and Potamitis (2017) sa zameriava na detekciu a segmentáciu spevu vtákov na otvorenom priestranstve. Vieme teda povedať, že sa jednalo o podobnú metódu na podobných dátach, ako tomu bolo v našej práci. Cieľom tejto práce bolo vytvoriť čo najviac automatizovaný program schopný vyhodnocovať veľký objem dát za účelom ich ďalšej analýzy. Od tej bola očakávaná identifikácia informácií o živote vtákov, napríklad analýza vplyvu klimatických zmien na život vtákov, presnejší odhad ich počtu, posudzovanie stavu ohrozených druhov a podobne. Na detekciu bolo postupne použitých niekoľko techník. Najprv *Salience* a *Class Activation Map* na vytvorenie tzv. *attention maps* (technológia, ktorá určí oblasť na obrázku, na ktorej sa najpravdepodobnejšie nachádzajú príznaky). Ďalej bola na presnú detekciu použitá neurónová sieť YOLOv2. Autori sa pre ňu rozhodli po porovnaní s *Single-Shot Detector* (skr. SSD) a *Faster R-CNN* vďaka tomu, že YOLOv2 preukázala najlepšie výsledky. Druhý použitý spôsob bol pomocou *U-net* architektúry (Ronneberger et al., 2015) a segmentácie spektrogramov. Na Obrázku 2 – 1 je možné vidieť výstup predikcie YOLO. Na vyhodnotenie metódy YOLO bola použitá metóda IoU, ktorá bola počítaná z predikcií a anotácií z *attention maps*. Najlepšia hodnota IoU bola 66,64%. Na vyhodnotenie metódy *U-net* bol použitý tzv. *Dice* koeficient, ktorý berie do úvahy pixely predikovaného objektu a pixely reálneho objektu z anotácií. Najvyššia hodnota koeficientu bola 74%.

Práca Konan et al. (2020) bola zameraná na doménu podobnú našej a zároveň ich problém bol riešený podobnou metódou. Jednalo sa totiž o analýzu bleskov, no iba ich určitej časti, nie každej udalosti na spektrogramoch. Táto časť, taktiež zobrazená na Obrázku 2 – 2, je v tejto práci nazvaná *whistler*. Presný čas, resp. frek-



Obrázok 2 – 1 Ukážka predikcie YOLOv2 na spektrogramoch. Zelenou farbou sú vyznačené anotácie, červenou predikcie (Fanioudakis and Potamitis, 2017).

vencia *whistlera* mala umožniť lepšie skúmanie plazmasféry (vnútorná magnetosféra, umiestnená pod ionosférou). Identifikácia a charakterizácia *whistlerov* sú preto dôležité úlohy na skúmanie plazmasféry. Z výsledkov tejto práce sa očakávalo vytvorenie ďalších databáz obsahujúcich detaily týchto udalostí, ktoré mohli byť ďalej použité pre štatistické analýzy.



Obrázok 2 – 2 Náhľad výstupu detekcie pomocou YOLO. Detegované sú 2 *whistlery* (Konan et al., 2020).

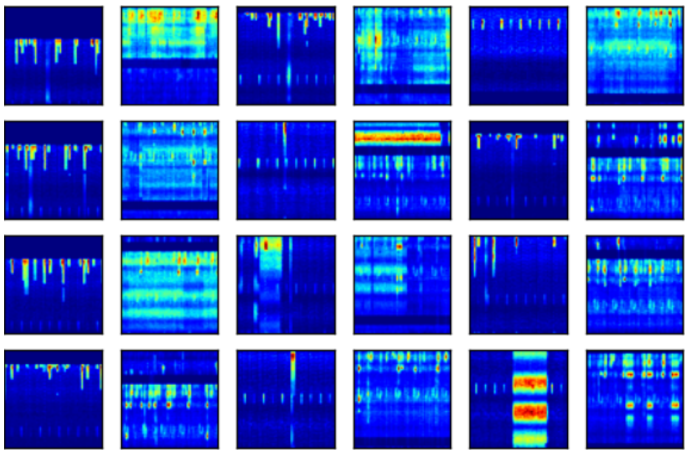
Daná práca porovnávala 3 rôzne detekčné modely. Jednalo sa o *Cross-Correlation with a Simulated Whistler* (skr. CCSW), *Sliding Deep Convolutional Neural Network* (skr. SDCNN) a už spomínanú *You Only Look Once* (skr. YOLO). Modely boli porovnávané na dvoch datasetoch z dvoch odlišných miest – ostrov Marion patriaci Južnej Afrike a výskumné stredisko SANAE IV na Antarktíde. Výsledky z trénovania boli prezentované porovnaním troch hodnôt – *Misdetection*, *False Alarm* a *F1 score* na datasete Marion. Pri porovnaní hodnôt v Tabuľke 2 – 1 môžeme skonštatovať, že v metrikách *Misdetection* a *F1 score* mal lepšie hodnoty model SDCNN. V metrike

False Alarm a časovom pomere detekcie mal lepšie výsledky model YOLO. Pomer času detekcie bol vypočítaný ako podiel času procesora na detekcii vzorky modelom a časovej dĺžky danej vzorky.

Model	Misdetection	False Alarm	F1 score	Časový pomer
CCSW	0,096	0,094	0,906	0,179
SDCNN	0,093	0,026	0,939	0,376
YOLO	0,141	0,021	0,915	0,010

Tabuľka 2 – 1 Porovnanie metrík troch použitých modelov na datasete Marion (Konan et al., 2020).

Publikácia O'Shea, Roy and Erpek (2017) je zasa zameraná na natrénovanie neurónovej siete schopnej nájsť a klasifikovať rádiové vlny na spektrogramoch podľa ich vlastností. Dataset bol vytvorený z reálnych dát a zachytával frekvencie typu GSM, LTE, ISM, FM a podobné bežne používané frekvencie. Na trénovanie bolo použitých celkom 8512 spektrogramov. Časť z nich je zobrazená na Obrázku 2 – 3.

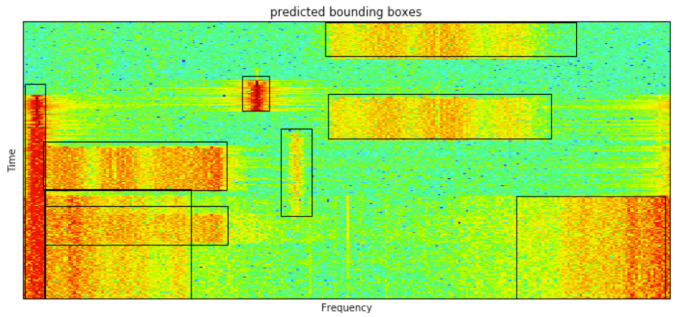


Obrázok 2 – 3 Výber spektrogramov použitých na trénovanie. Jedná sa o spektrogramy rádiových vĺn typu 802.11b/g/n, Bluetooth, FD-LTE, QAM a podobne (O'Shea, Roy and Erpek, 2017).

Ako technológia bola zvolená neurónová sieť typu VGG. Priemerná presnosť klasifikácie bola po natrénovaní 0,944. Presnosť klasifikácie sa s ohľadom na túto

metriku javí ako veľmi dobrá, rovnako ako čas, za ktorý je sieť schopná vyhodnotiť jeden spektrogram. Ten bol v priemere 0,3 ms.

Poslednou popísanou publikáciou v tejto analýze je O'Shea, Roy and Clancy (2017). Zaoberala sa lokalizáciou rádiových signálov na spektrogramoch. Autori sa rozhodli použiť YOLOv2, ktorú porovnávali s *Faster RCNN* a *Single Shot Multibox Detector*. Vzhľadom na výkonnostné požiadavky a rýchlosť, ktorou vie daná neurónová sieť vyhodnocovať obrázky, bolo YOLO jasné rozhodnutie. Trénovanie bolo vykonávané na 20000 spektrogramoch, z ktorých každý obsahoval aspoň jednu udalosť. Výstup predikcie je možné vidieť na Obrázku 2 – 4. Natrénovaný model dokázal predikovať spektrogramy rýchlosťou približne 3 ms na obrázok aj na menej výkonných zariadeniach, čo bohato postačuje na predikcie aj v reálnom čase.



Obrázok 2 – 4 Predikcie YOLO neurónovej siete na spektrograme rádiového pásma ISM (O'Shea, Roy and Clancy, 2017).

3 Automatická detekcia elektromagnetických pulzov metódami hlbokého učenia

Všetky údery bleskov generujú elektromagnetické pulzy, ktoré dokážu prekonávať vzdialenosti tisícov kilometrov. Vďaka dnešným technológiám dokážeme tieto údery zaznamenať v rádiovej oblasti do 30 kHz veľmi nízkej frekvencie (angl. *very low frequency*, skr. VLF). Ich analýzou je možné nielen lokalizovať úder blesku, ale aj charakterizovať vlastnosti prostredia medzi bleskom a detektorom. Pri našej práci sme používali spektrogramy, na ktorých boli dané pulzy zachytené. Zdrojom týchto spektrogramov bolo zariadenie ELMAVAN-G (Santolík and Kolmašová, 2017). Názov ELMAVAN-G je skratka pre elektromagnetický vlnový analyzátor. Tento prístroj je vyvíjaný Ústavom fyziky atmosféry Akadémie vied Českej republiky (skr. AV ČR) pre družicový projekt *Resonance*. Prístup k dátam meracieho prístroja ELMAVAN-G sme mali na základe spolupráce so Slovenskou akadémiou vied, ktorá spolupracuje s Ústavom fyziky atmosféry AV ČR.

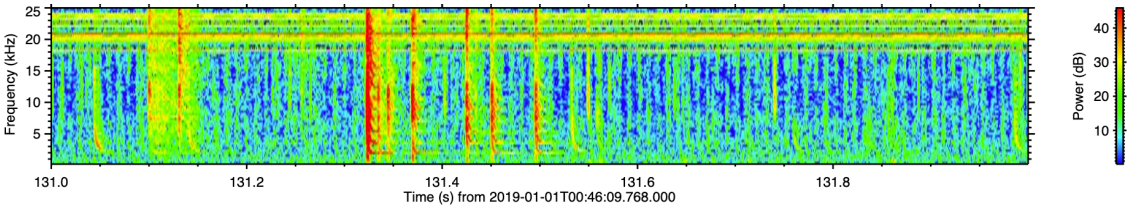
Obrázok 3 – 1 Jedna z troch častí VLF analyzátora ELMAVAN-G určeného na príjem signálov vo frekvencii do 25 kHz. Zdroj: `http://bleska.ufa.cas.cz/lsbb/pics_s/vlf_big.jpg`

Analyzátor pozostáva z troch častí. Konkrétne z dvoch magnetických slučkových antén s 12 závitmi, každý s plochou 4 m^2 a jednej guľovej elektrickej antény s prie-

merom 10 *cm*, ktorá je umiestnená 2 *m* nad zemou. Na Obrázku 3 – 1 je možné vidieť jednu zo slučkových antén. Analyzátor sa nachádza na vrchole hory La Grande Montagne (1028 *m* n. m.) neďaleko mesta Rustrel vo Francúzsku. Patrí pod laboratórium *Laboratoire Souterrain a Bas-Bruit* a vďaka spolupráci Ústavu fyziky atmosféry AV ČR s touto organizáciou sme mali k dispozícii dostatok dát z týchto meraní (Santolík and Kolmašová, 2017).

3.1 Pochopenie cieľa

Základným cieľom tejto práce je vytvorenie funkčnej neurónovej siete, ktorá bude schopná analyzovať jednotlivé spektrogramy a lokalizovať v nich bleskové výboje. Príklad spektrogramu použitého v tejto práci je možné vidieť na Obrázku 3 – 2. Bleskové výboje sa v spektrogramoch nazývajú sfériky a sú rozoznateľné ako vertikálne čiary zasahujúce celý frekvenčný rozsah alebo jeho veľkú časť. Tieto čiary by mala naša neurónová sieť vedieť nájsť. Ako výstup práce bola požadovaná tabuľka pozostávajúca z informácií o daných bleskových výbojoch. Tabuľku sme vytvorili spracovaním výstupov neurónovej siete a následným dopočítaním ďalších požadovaných dát. Celý priebeh bude popísaný v nasledujúcich podkapitolách.



Obrázok 3 – 2 Náhľad celého spektrogramu spolu s frekvenčným rozsahom umiestneným na pravej strane.

Tabuľka by mala pozostávať zo stĺpcov:

1. `image #`

   Stĺpec `image #` vyjadruje poradové číslo anotovaného spektrogramu.

2. `event #`

StĺpecTable `event #` vyjadruje poradové číslo lokalizovanej udalosti[7] na spektrograme.

3. `date`

   StĺpecTable `date` vyjadruje dátum a čas merania vo formáte `YYYYMMDD_HHMMSS`. Od tohto údaja sa počítali sekundy a milisekundy nachádzajúce sa v ďalších stĺpcoch.

4. `second`

   Sekunda, ktorej prislúcha daný *event*. Počíta sa od času `date`.

5. `milisecond`

   Milisekunda, v ktorej sa *event* udial. Jedná sa o presný časový údaj určený metódou automatického dopočítania stredu *eventu* na základe intenzity červenej farby jednotlivých pixelov. Ako údaj sa berie pozícia, v ktorej je daný *event* najvýraznejší.

6. `tweek`

   StĺpecTable pracujúci s výstupom neurónovej siete, ktorý bude obsahovať hodnoty 0 pre stav, kedy daný sférik neobsahuje disperzie (nie je to *tweek*) a 1 kedy ich obsahuje (je to *tweek*).

7. `f_min<2kHz`

   StĺpecTable určený automatickou metódou segmentácie pracujúci so skutočnosťou, či emisia daného *eventu* zasahuje do priestoru dolných 2 kHz spektrogramu alebo nie. V prípade, že nezasahuje bude stĺpec obsahovať hodnotu 0, ak zasahuje, tak 1.

Aby sme sa dopracovali k cieľu, potrebovali sme prejsť nasledujúcimi krokmi:

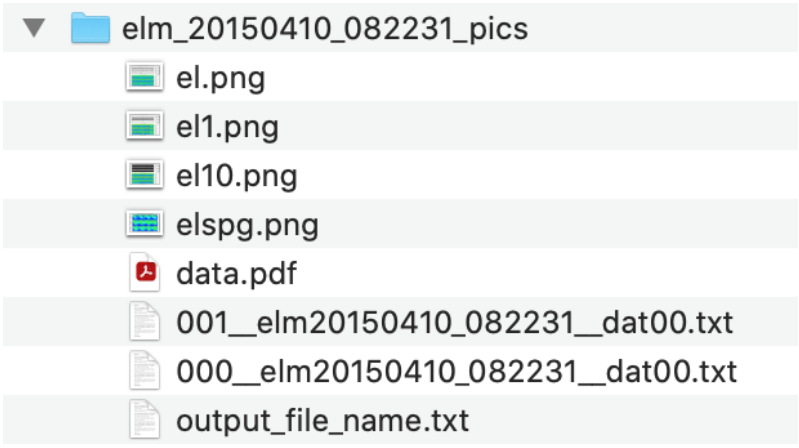
[7] V ďalšom texte budeme používať anglický výraz *event* s cieľom odlíšiť a vystihnúť jedinečnosť udalostí.

- Stiahnutie dát na lokálne úložisko a ich pochopenie,
- spracovanie dát,
- vytvorenie anotačného projektu a priebeh anotácií,
- spracovanie anotačného projektu a vizualizácia anotácií ako jeho výstupov,
- nájdenie vyhovujúcej neurónovej siete,
- spracovanie anotovaných dát na dataset, trénovanie siete na datasete a následné vyhodnotenie metrík,
- spracovanie výsledkov neurónovej siete a získanie ďalších dát automatickými metódami dopočítania a segmentácie,
- zapracovanie dát do tabuľky.

V tomto bode bolo potrebné si stanoviť merateľné kritéria úspešnosti (KPI). V našom prípade bola dôležitá najmä vizuálna kontrola, nakoľko rozmery boxov anotácií ohraničujúcich *eventy* neboli optimálne. Rovnako často sa stávalo, že boli sfériky veľmi blízko seba, čo anotátorom znemožnilo ich dôkladné označenie. Práve preto sme stanovili nižšie hodnoty KPI pre detekciu objektov. Na základe konzultácie s doménovými expertmi sme sa rozhodli pre hodnotu 65 % pre vážený priemer metriky F1 skóre. Merateľný cieľ určenia presnej milisekundy nebolo možné určiť, nakoľko sú tieto udalosti vyhodnocované len na základe vizuálnej kontroly. Skutočnú hodnotu milisekundy nepoznáme, vieme však na základe anotácií vyhodnotiť veľmi dôležitý ukazovateľ pre ďalší výskum, ktorým je fakt, či emisia jednotlivých *eventov* siaha pod 2 kHz alebo nie. Pre tento bod sme stanovili cieľ dosiahnuť vážený priemer pre metriku F1 skóre nad 90 %.

3.2 Pochopenie dát

Na prácu na tomto projekte nám boli poskytnuté dáta od AV ČR. Jednalo sa o výstupy zo zariadenia ELMAVAN-G[8]. V našej práci sme pracovali konkrétne s dátami z obdobia rokov 2015 až 2019. Vďaka tomu, že sa jednalo o pravidelné merania v 5 minútových intervaloch, bolo dát na trénovanie dostatok. Dáta z každého merania boli uložené v dvoch archívoch – `tar.bz2` a `zip`. Oba archívy mali podobný základ názvu, ktorý pozostával zo slova *elm*, ktoré znamená skratku pre zdroj dát, dátumu a času merania v tvare `RRRRMMDD_HHMMSS`. V prvom spomínanom archíve sa nachádzali logovacie súbory s textovými a binárnymi výstupmi z analyzátora. V archíve `zip`, ktorý mal pre odlíšenie na konci názvu slovo *pics*, sa nachádzalo niekoľko textových súborov s výstupom analyzátora a 4 obrazové súbory typu `png`. V týchto sa nachádzal prehľad všetkých udalostí a vyobrazenie najsilnejšej a najslabšej emisie. Ako posledný bol v archíve uložený súbor `data.pdf`, ktorý obsahoval súhrnný obrazový výstup z každého merania. S týmto archívom sme pracovali aj my v tejto práci. Štruktúru archívu s obrazovým výstupom, ktorý bol vytvorený po každom meraní je možné vidieť na Obrázku 3 – 3.



Obrázok 3 – 3 Štruktúra priečinka, ktorý bol vytvorený po každom meraní obsahujúca obrazový výstup.

[8]V čase písania práce boli dáta dostupné na odkaze `http://bleska.ufa.cas.cz/lsbb/storage/elm/`.

Vstupné dáta, s ktorými sme pracovali mali podobu spektrogramov. Spektrogramy sú názorné grafy pozostávajúce z osí X a Y. V našom prípade os X zodpovedala času jednej sekundy. Na osi Y sa nachádzala frekvencia merania, ktorá bola v rozsahu 0 kHz – 25 kHz. Merania boli vykonávané v 5 minútových časových intervaloch. Doba každého merania bola 144 sekúnd. Z jednotlivých meraní analyzátor vyhodnotil 10 obrázkov, 9 z nich poukazovalo na sekundy merania s najvyššou emisiou, desiaty obrázok poukazoval na najnižšiu emisiu. To znamená, že za každých 24 hodín merania je dostupných takmer 2600 obrázkov, čo je za 1 rok asi 950000. Ide teda skutočne o veľké množstvo dát, ktoré nie je možné analyzovať bez techník počítačového videnia. A práve s obrázkami, na ktorých boli zachytené najvyššie emisie sme pracovali v tejto práci.

3.3 Príprava dát

Túto časť sme zhodnotili ako najzložitejšiu z celej práce, keďže obnášala pochopenie pre nás nových dát a problematiky, ich následnú analýzu a navrhnutie správnej počiatočnej úpravy na ďalšie použitie v anotačnom projekte.

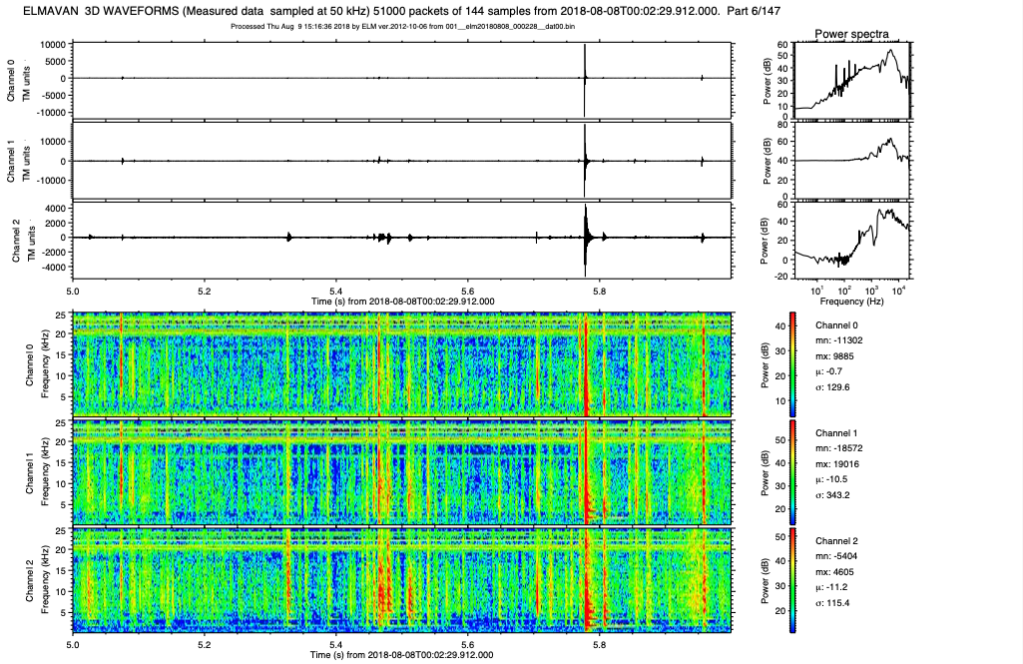
3.3.1 Sťahovanie a spracovanie dát

Vzhľadom na to, že boli dáta uložené na vzdialenom úložisku, ku ktorému sme mali prístup iba na čítanie, bolo pre ďalšiu prácu nevyhnutné tieto dáta dostať na nám prístupné lokálne úložisko, aby s nimi bolo možné manipulovať. Na tieto potreby nám bol sprostredkovaný dátový priestor patriaci Technickej univerzite v Košiciach o veľkosti približne 16 TB a Linux server, ktorý postačoval na naše potreby. Následne sme pomocou príkazu `wget` stiahli všetky dáta zo serverov AV ČR na školské úložisko. Po stiahnutí nasledoval ďalší krok – predpríprava dát na anotačný projekt. Pre tieto potreby sme vytvorili komplexný skript pozostávajúci z viacerých krokov, ktorý bude dostupný k náhľadu na online github repozitári tejto práce[9]. V prvom

[9] `https://github.com/space-lab-sk/tweeks_detection`

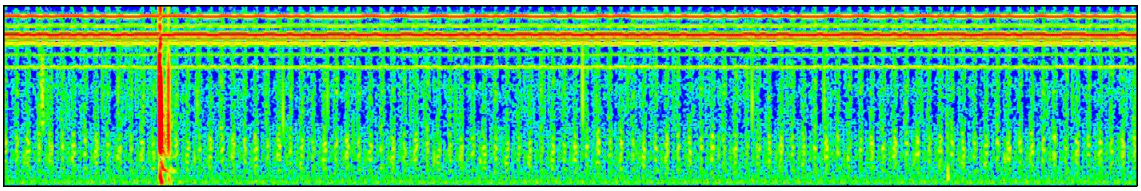
kroku sme rozbalili všetky archívy obsahujúce obrázky, a teda archívy typu `zip` do rovnomenného priečinka. Následne sme z novovytvorených priečinkov vymazali všetky nepotrebné súbory. Krok vymazania bol uskutočnený z dôvodu prehľadnosti a nižšieho zahlcovania disku malými súbormi. Po tomto kroku nám v každom priečinku ostal iba súbor `data.pdf`, s ktorým sme ďalej pracovali. Každý `data.pdf` súbor pozostával z 11 strán. Prvá strana bola súhrnná, čiže celých 144 sekúnd merania v jednom spektrograme. Nasledovalo 9 strán obsahujúcich 9 najsilnejších sekúnd. Posledná strana súboru `data.pdf` zobrazovala sekundu merania s najslabšou emisiou. Zo súboru `data.pdf` sme teda vybrali strany obsahujúce najsilnejšie sekundy, rozdelili ich na 9 nových súborov a konvertovali ich formát na `jpg`. Do ich názvu sme okrem vyššie spomenutého časového údaja, ktorý bol aj názvom priečinka, zakomponovali aj údaj o presnej stotine merania a taktiež údaj o sekunde, ktorý sa nachádzal na každej strane. Tento údaj sme boli schopní dostať pomocou parsovania textu v dokumente. Po tomto kroku sme v každom priečinku dostali 9 nových súborov `jpg`, ktorých náhľad je dostupný na Obrázku 3 – 4.

Každý z obrázkov, s ktorými sme pracovali, pozostával z viacerých častí. Prvá časť obrázka umiestnená na hornej polovici obsahovala tri panely – jeden znázorňujúci vlnové formy vertikálneho elektrického poľa a dva znázorňujúce horizontálne zložky magnetického poľa podľa antén analyzátora ELMAVAN-G. Dolná polovica obrázkov obsahovala tri panely znázorňujúce časovo-frekvenčné spektrogramy počítané z nameraných vlnových foriem. Tieto spektrogramy boli pre nás dôležité a potrebovali sme ich extrahovať do takej podoby, aby boli ďalej použiteľné na anotáciu a potom ako vstup na učenie neurónovej siete. Po konzultácii s doménovými expertmi z AV ČR sme sa rozhodli, že pre naše potreby použijeme spodný – tretí panel, pretože sa na vybraných príkladoch javil ako najvýraznejší – udalosti na ňom boli najlepšie vidieť. Ďalšie rozhodnutie, ktoré sme na konzultácii učinili bolo, že na anotovanie a trénovanie neurónovej siete budeme používať iba spodných 18 kHz z každého spektrogramu. Dôvod tohto rozhodnutia bol, že sa na každom spektrograme na pozícii 18 kHz – 25 kHz nachádzali horizontálne čiary, ktoré je možné



Obrázok 3 – 4 Náhľad jednej zo strán súboru `data.pdf`, ktorý obsahoval výsledky merania.

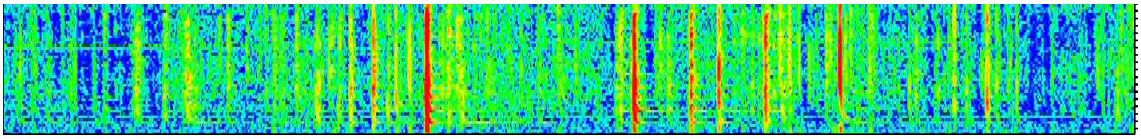
vidieť na Obrázku 3 – 5. Tieto čiary boli signály z nízkofrekvenčných výkonných vysielačov používaných na komunikáciu s ponorkami a nijako nesúviseli so sledovanými udalosťami. Na základe tejto skutočnosti sme sa rozhodnutím oddeliť túto časť spektrogramov snažili predísť chybám pri učení neurónovej siete.



Obrázok 3 – 5 Spektrogram obsahujúci celých 25 kHz, vrátane signálov vysielačov, ktoré nesúviseli s bleskami.

Finálne rozhodnutie na orezávanie obrázkov bolo nasledovné – z každej najsilnejšej sekundy merania sme použili jeden spektrogram. Jednalo sa o spektrogram zo spodného panelu (*channel* 2), ktorý ale obsahoval iba spodných 18 kHz. Konečné

spektrogramy mali rozmery 1421 × 162 pixelov. Vyobrazenie príkladu konečného spektrogramu je dostupné na Obrázku 3 – 6. V ďalšej podkapitole bude objasnený proces vytvorenia anotačného projektu, na vstupe ktorého boli takto pripravené obrázky.



Obrázok 3 – 6 Spektrogram obsahujúci finálnych 18 kHz, bez signálov vysielačov. Takýto typ spektrogramov bol použitý na anotácie a trénovanie neurónovej siete.

3.3.2 Realizácia anotačného projektu na platforme Zooniverse

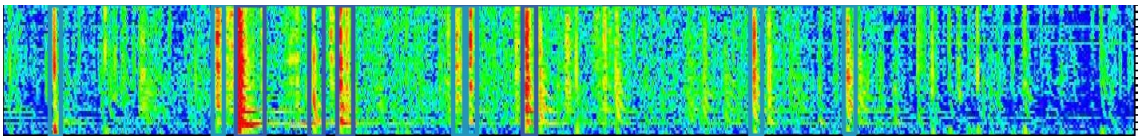
Pri detekcii objektov pomocou neurónových sietí sa na trénovanie používajú vstupné dáta v podobe anotovaných datasetov. Anotované datasety sú množiny dát s vyznačenými udalosťami, ktoré by mala neurónová sieť vedieť nájsť. Dané vyznačenia sú reprezentované mnohými špecifickými spôsobmi, v našom prípade sa jednalo o textový súbor s rovnomenným názvom ako obrázok a vloženými súradnicami daných *eventov*. Vzhľadom na to, že v tejto časti sme disponovali množinou obrázkov bez anotácií, vytvorili sme anotačný projekt na platforme *Zooniverse*, pomocou ktorého sme dostali crowdsourcingovou metódou anotácie pre našu neurónovú sieť. Anotačný projekt bol vybraný na základe skutočnosti, že neurónová sieť potrebuje na čo najpresnejšie trénovanie vysoký počet unikátnych obrázkov na vstupe. V našom prípade sa jednalo o približne 2500 obrázkov. Vzhľadom na to, že označiť takýto počet obrázkov by jednému človeku trvalo dlhý čas, sme sa rozhodli pre tento účel vytvoriť takýto projekt. S anotáciami nám pomohli študenti Technickej univerzity v Košiciach.

V prvom rade sme potrebovali vytvoriť anotačný projekt[10] na platforme *Zooniverse* zameraný na anotovanie obrázkov. K nemu bolo nevyhnutné vytvoriť pod-

[10]Anotačný projekt je dostupný na odkaze `https://www.zooniverse.org/projects/akundrat/anotacia-bleskov`. Anotovanie bolo rozdelené do dvoch častí. V jednej časti bolo ozna-

robný tutoriál, aby študenti jasne pochopili, čo budú na obrázkoch hľadať. V ňom bolo zahrnuté stručné vysvetlenie dát a presný popis spôsobu anotovania vrátane správnych a nesprávnych príkladov anotácií. Aj vďaka podrobnému tutoriálu sme pri anotáciách predišli nepochopeniu a tým pádom sa eliminoval počet nesprávne anotovaných obrázkov.

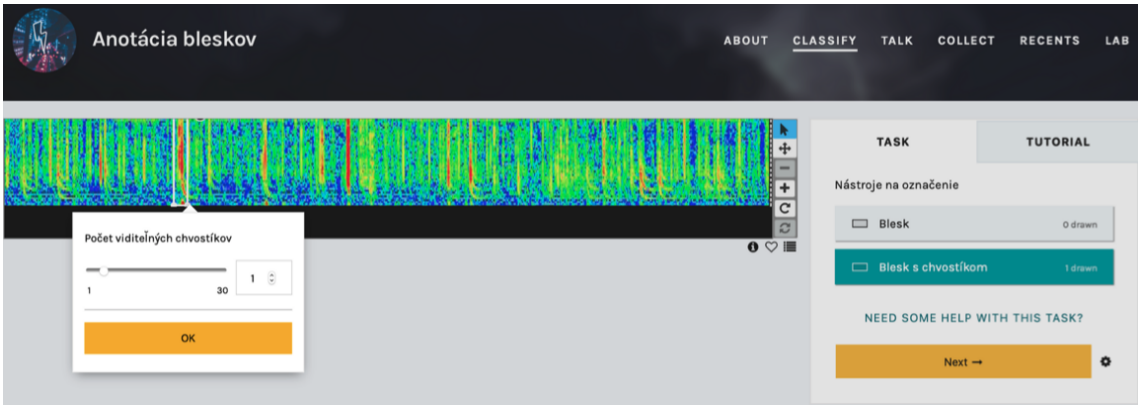
Po vytvorení tutoriálu sme pokračovali vytvorením anotačných nástrojov, ktoré by sme mohli nazvať základnou funkcionalitou anotačného projektu. Anotačné nástroje boli dva. Prvým bolo možné označiť bežný sférik, na Obrázku 3 – 7 označený rámčekom so svetlomodrou farbou. Druhým boli označované sfériky s disperziou, tzv. *tweeky*, ktoré sú na Obrázku 3 – 7 označené rámčekom s tmavomodrou farbou. Zároveň pri výbere označenia *tweeku* bol od anotátora vyžiadaný údaj o počte disperzií, teda tzv. chvostíkov daného *eventu*, ako je možné vidieť na Obrázku 3 – 8. Posledná časť základu anotačného projektu bol druhý, stručnejší tutoriál, ktorý bol dostupný hneď pri anotačných nástrojoch. Tento tutoriál slúžil na rýchle objasnenie použitia anotačných nástrojov v prípade, že by si anotátor nebol istý svojim rozhodnutím.



Obrázok 3 – 7 Anotovaný obrázok obsahujúci označené *eventy*.

Po vytvorení anotačného projektu sme ho naplnili obrázkami. Obrázky boli vyberané náhodne. Rozhodli sme sa pre výber iba zo štyroch najteplejších mesiacov v roku (júl – september), keďže to je obdobie, kedy je búrková aktivita v Európe najintenzívnejšia. V každom z dostupných rokov bol vybraný jeden mesiac a tieto obrázky boli naplnené do anotačného projektu. Po tomto kroku bolo možné začať s anotáciami.

čených približne 1400 obrázkov. Po ukončení prvej časti anotácií bola táto časť datasetu stiahnutá z anotačného projektu pre lepší prehľad. V druhej časti bolo označených zvyšných približne 1100 obrázkov. Tieto obrázky sú stále dostupné v anotačnom projekte.



Obrázok 3 – 8 Náhľad anotačného nástroja s označeným *tweekom*. V tomto stave ešte bolo potrebné doplniť počet disperzií.

Anotovanie bolo uskutočnené študentmi Fakulty elektrotechniky a informatiky Technickej univerzity v Košiciach. Ako výstup z anotovania sme dostali viac ako 22000 *eventov* označených na približne 2500 obrázkoch. Počet *tweekov* na výstupe anotačného projektu bol necelých 3200, vo zvyšných prípadoch sa jednalo o sfériky. Nasledovalo testovacie vykreslenie približne stovky obrázkov. Keďže boli informácie o anotáciách uložené v separátnom súbore typu `json`, museli sme najprv vytvoriť skript, ktorý ich dokázal spracovať a na základe ich obsahu vykresliť označujúce obdĺžniky priamo do obrázkov. Na základe manuálnej kontroly tohto testovacieho vykreslenia sme skonštatovali, že nie všetky anotácie sú presné. Zhodli sme sa, že bude potrebné vykresliť a manuálne prekontrolovať všetky obrázky, aby nemohla nastať situácia, kedy by kvôli nesprávnym anotáciám dochádzalo k chybovosti učenia sa a vyhodnocovania neurónovej siete. Po vykreslení a kontrole všetkých obrázkov sa nám ich počet zredukoval na necelých 2300. S týmto datasetom sme mohli ďalej pracovať.

3.3.3 Normalizácia a rozdelenie dát

Pred začatím trénovania sme potrebovali vytvoriť dataset obrázkov v správnom tvare pre potreby zvolenej neurónovej siete YOLOv5. Na trénovanie modelu na vlastnom datasete, YOLOv5 požaduje okrem množiny obrázkov taktiež ku každému obrázku

rovnomenný textový súbor obsahujúci informácie o nich. Konkrétne sa jedná o 5 údajov o každom *evente*.

```
1 0.96309 0.5 0.01714 1
1 0.88881 0.5 0.01224 1
1 0.8227 0.5 0.01224 1
1 0.75617 0.5 0.00816 1
1 0.70067 0.5 0.01143 1
0 0.64639 0.5 0.00735 1
0 0.57823 0.5 0.01143 1
```

Obrázok 3 – 9 Náhľad textového súboru obsahujúceho normalizované anotácie ku jednému zo spektrogramov.

Na Obrázku 3 – 9 je zobrazený náhľad jedného z textových súborov ku obrázku. Ako oddeľovač bola v týchto súboroch použitá medzera. Pre jednoduchšie vysvetlenie vieme každý z riadkov rozdeliť do troch skupín. Prvú skupinu reprezentuje prvý údaj. Jedná sa o triedu (angl. *class*). Tento údaj reprezentoval, o aký druh *eventu* sa jedná. V našom prípade trieda 0 reprezentovala sférik a trieda 1 reprezentovala *tweek*. Do druhej skupiny patria ďalšie dva údaje – čiže stĺpec č. 2 a 3 na Obrázku 3 – 9. Tieto údaje poukazujú na súradnice stredu daného obdĺžnika, ktorý vznikol anotáciou. Konkrétne stĺpec č. 2 reprezentuje hodnotu X a stĺpec č. 3 patrí hodnote Y. Zvyšné dva stĺpce reprezentujú šírku a výšku obdĺžnika, presne vo vyslovenom poradí. Okrem prvej hodnoty sú všetky tieto údaje normalizované, to znamená že sú z intervalu medzi 0 a 1. Normalizované dáta boli jednou z požiadavok na vstupe do neurónovej siete kvôli univerzálnosti, keďže rozlíšenie obrázka na vstupe neurónovej siete nemusí byť vždy rovnaké, ako jeho natívne rozlíšenie. Normalizované hodnoty sme dostali predelením natívnych hodnôt celkovým rozlíšením danej strany. Pre lepšiu predstavu je na Rovnici 3.1 priblížený proces vypočítania normalizovanej šírky obdĺžnika, kde hodnota *width_box* zastupuje reálnu šírku obdĺžnika anotovaného blesku v pixeloch a za hodnotu *width_picture* sme dosadili celkovú šírku spektrogramu, v našom prípade 1421 pixelov. Podobným spôsobom sme vypočítali aj normalizovanú výšku – jednalo sa o pomer výšky boxu a výšky obrázka v pixeloch.

$$\text{width\_norm} = \frac{\text{width\_box}}{\text{width\_picture}}. \tag{3.1}$$

Po takejto príprave dát sme potrebovali pred naplnením neurónovej siete rozdeliť dataset na 3 množiny – trénovaciu, validačnú a testovaciu. Na trénovacej množine sa neurónová sieť učí, na validačnej množine je možné nastavovať hyperparametre. Používa sa na optimalizáciu a regularizáciu. Ku testovacej množine nemá neurónová sieť počas trénovania vôbec prístup, uvidí ju prvýkrát až pri spustení testovania. Predchádza sa tým skutočnosti, aby vstupy v testovacej množine neovplyvňovali trénovanie a tým pádom nemohlo dochádzať k ovplyvneniu výsledkov. Na začiatku sme z množiny 2300 spektrogramov vybrali približne 230 obrázkov, ktoré sme označili ako testovaciu množinu. Tým pádom môžeme povedať, že pomer trénovacích obrázkov ku testovacím bol `9:1`. Zo zvyšných približne 2000 obrázkov sme vytvorili trénovaciu a validačnú množinu. Na vytvorenie validačnej množiny sme oddelili ďalších približne 400 obrázkov a zvyšných 1600 ostalo na trénovanie. Validačná množina slúži na reguláciu chyby pri trénovaní a používa ju neurónová sieť automaticky, nie je tam potrebný zásah užívateľa.

3.4 Modelovanie

Prípravu na trénovanie modelu sme v predchádzajúcom kroku ukončili a preto sme mohli prejsť ďalej ku spusteniu trénovania. Trénovanie bolo spustené nasledovným príkazom, ktorého parametre sú popísané nižšie:

```
!python train.py --img 1184 --rect --batch 4 --epochs 400 --data
./data/blesky.yaml --cfg ./models/yolov5s.yaml --weights " --device 0
```

- `!python`

  Časť volajúca rodičovský *Python* súbor, pomocou ktorého na termináli nastavíme, akým spôsobom sa bude spúšťať daný príkaz.

- `train.py`

Časť odkazujúca sa na skript, ktorý spúšťa trénovanie modelu.

- `--img 1184`

  Veľkosť vstupného obrázka v pixeloch. Používali sme nižšiu veľkosť ako bola reálna, kvôli nedostatočnému výpočtovému výkonu trénovacieho hardvéru.

- `--rect`

  Parameter zabezpečuje, aby sa neurónová sieť pozerala na trénovacie dáta ako na obdĺžnik, nie ako na štvorec. Ak by sme nepoužili tento parameter, neurónová sieť by z našich obrázkov s rozmermi 1421×162 spravila štvorce o rozmere 1184×1184 a trénovanie by prebiehalo na takýchto obrázkoch.

- `--batch 4`

  Hodnotu hyperparametra *batch size* sme zvolili 4, čo znamená, že pri trénovaní prechádzajú vždy kombinácie 4 obrázkov cez neurónovú sieť v reálnom čase[11]

- `--epochs 400`

  Počet epoch znamená číslo, ktoré vyznačuje, koľko krát prejdú všetky trénovacie záznamy neurónovou sieťou. V našom prípade sme zvolili 400 epoch s vedomosťou, že je model schopný regularizácie a pracuje s regularizačnými technikami *checkpoint* a *early stopping*.

- `--data ./data/blesky.yaml`

  Cesta ku `yaml` súboru, v ktorom boli uložené detaily o trénovacej, testovacej a validačnej množine a taktiež o triedach.

- `--cfg ./models/yolov5s.yaml`

  Súbor `yaml` s odkazom na model, pomocou ktorého prebiehalo trénovanie. My sme trénovali s najnižším modelom, čiže jeho označenie bolo `S`.

[11]Nakoľko je model YOLOv5 výpočtovo náročný, zvolená hodnota bola maximálnou pri využití grafickej karty Quadro RTX 4000, ktorú sme na trénovanie používali. Technické špecifikácie sú dostupné na `https://www.nvidia.com/en-us/design-visualization/quadro/rtx-4000/`.

- `--weights ' '`

  Do tohto parametra je možné vložiť odkaz na váhy, keď prebieha ladenie modelu a teda keď sa nejedná o prvé trénovanie. Keďže my sme trénovali úplne od nuly, ostal tento parameter prázdny. V prípade, že užívateľ používa YOLO na klasifikovanie objektov bežného sveta, je možné do tohto parametra vložiť odkaz na existujúce váhy vytvorené vývojármi.

- `--device 0`

  Možnosť vybrať, na ktorej konkrétnej grafickej karte bude trénovanie prebiehať v prípade, že nimi zariadenie disponuje.

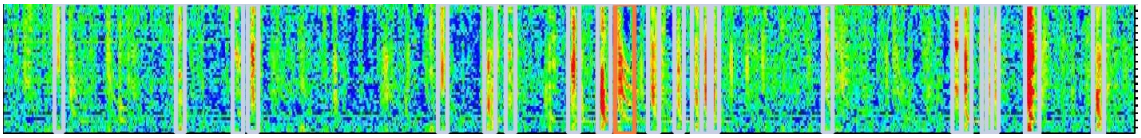
Po trénovaní trvajúcom niekoľko hodín sme boli schopní spustiť detekciu na oddelenej množine obrázkov, ktoré sa do neurónovej siete pri trénovaní nedostali. Na detekciu sme použili súbor `detect.py`, po spustení ktorého nám program vygeneroval predikcie na množine obrázkov, ktoré dostal na vstupe. Čo sa týka parametrov, bolo možné zvoliť parameter *Intersection over Union* (skr. IoU) a *confidence threshold*. Metrika IoU v tomto prípade porovnávala boxy samotných tried pri jednotlivých predikciách. Jej hodnota sa počíta podľa Vzťahu 3.2, kedy sa pracuje s pomerom prieniku jednotlivých boxov a zjednotenia týchto boxov.

$$\text{IoU} = \frac{\text{prienik}}{\text{zjednotenie}}. \tag{3.2}$$

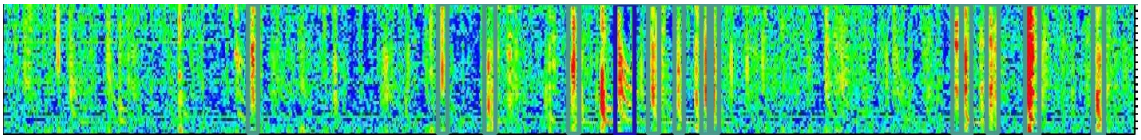
Metrika *confidence threshold* zasa pracuje s hodnotou pravdepodobnosti zaradenia do danej triedy neurónovej siete, kedy platná predikcia je iba tá, ktorá presiahne túto hodnotu. Pri našej predikcii sme, ako je možné vidieť na nasledujúcom príkaze, používali tieto dve metriky ako desatinné vyjadrenie percentuálnej hodnoty.

```
!python detect.py --source ./test_img --weights runs/train/exp19/
weights/best.pt --iou 0.45 --conf 0.3 --img 1184 --save-txt
```

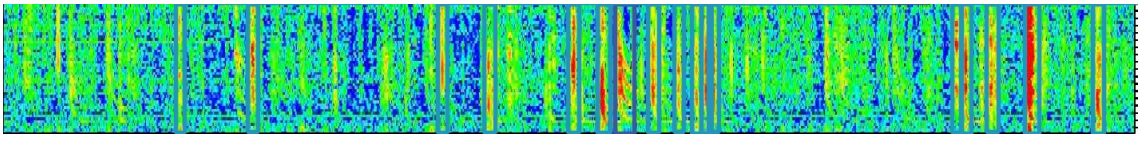
Pri spúšťaní predikcií sme pracovali s viacerými hodnotami parametrov IoU a *confidence*, kým sme našli tú optimálnu. Pri hľadaní správnej hodnoty IoU sme sa odrážali od základnej hodnoty modelu, keďže pri nej sme dosahovali najlepšie výsledky predikcií. Pri hľadaní správnej hodnoty *confidence thresholdu* sme taktiež prechádzali mnohými možnosťami. Táto hodnota nám pomáhala regulovať akúsi pomyselnú citlivosť detektora, kedy nám pri veľmi nízkom *thresholde* aplikácia nachádzala prehnane vysoké množstvo *eventov* (k náhľadu na Obrázku 3 – 10). Na tomto obrázku môžeme vidieť detekciu modelu s parametrom *conf* na hodnote 0,1. Na druhej strane pri veľmi vysokej hodnote nám model nachádzal neprimerane nízke množstvo *eventov* (k náhľadu na Obrázku 3 – 11). Hodnota spomínaného parametra pri tomto obrázku bola 0,45. Optimálne zvolená približná hodnota *conf* na základe viacerých testov teda nakoniec bola 0,30. Predikciu s touto hodnotou je možné vidieť na Obrázku 3 – 12. Dolaďovanie danej hodnoty je popísané podkapitole 3.6.



Obrázok 3 – 10 Náhľad predikcie pri hodnote IoU 0,45 a *conf* 0,1.



Obrázok 3 – 11 Náhľad predikcie pri hodnote IoU 0,45 a *conf* 0,45.



Obrázok 3 – 12 Náhľad predikcie pri hodnote IoU 0,45 a *conf* 0,30.

3.5 Post-processing

V tejto podkapitole je popísané, akým spôsobom boli dáta z predikcií spracované, aby na výstupe bola požadovaná tabuľka informácií a taktiež priebeh práce s obrázkami za účelom vytvorenia ďalších stĺpcov vo výstupnej tabuľke.

Pri časových stĺpcoch (`date` a `second`) nám na ich vytvorenie postačovala práca s textovými údajmi, kedy tieto dáta boli spracované z časti názvu súboru, o ktorý sa jednalo. Pri stĺpcoch `milisecond` a `f_min<2kHz` ich ale bolo potrebné vytvoriť, keďže takéto údaje sme na výstupe neurónovej siete nedostali a taktiež ich nebolo možné dostať ani parsovaním pôvodného dokumentu. Priebeh ich dotvárania je popísaný v nasledujúcom oddiele. Stĺpec *tweek* sme vytvorili na základe výstupu neurónovej siete, ktorá svoje detekcie delila do dvoch tried. Tieto dve triedy nám na výstupe slúžili na vyhodnotenie, či sa jedná o sférik alebo o *tweek*. V stĺpci sme túto skutočnosť vyjadrili číslom 0 alebo 1, ktoré vyjadrovalo, či je daný stĺpec *tweek* (pre hodnotu 1) alebo ním je sférik (pre hodnotu 0). Náhľad tabuľky aj s názvami stĺpcov si je možné pozrieť na Obrázku 3 – 13.

image	event	date	second	milisecond	tweek	f_min<2kHz
8	1	20150902_231933	102	100	1	1
8	2	20150902_231933	102	129	1	1
8	3	20150902_231933	102	153	0	1
8	4	20150902_231933	102	190	1	1
8	5	20150902_231933	102	246	1	1
8	6	20150902_231933	102	417	0	0
8	7	20150902_231933	102	786	0	1
8	8	20150902_231933	102	813	0	1
8	9	20150902_231933	102	864	0	1
8	10	20150902_231933	102	913	0	0
8	11	20150902_231933	102	966	0	1

Obrázok 3 – 13 Obrázok tabuľky vytvorenej na konci programu, ktorá obsahuje údaje o jednom spektrograme.

3.5.1 Segmentácia

Tento oddiel popisuje spôsob, akým sme nadobudli údaje do stĺpcov `milisecond` a `f_min<2kHz`. Na vytvorenie týchto stĺpcov sme potrebovali vytvoriť nový skript,

ktorý s využitím matematických výpočtov a segmentačných techník dôjde k výsledku, ktorý požadujeme. Zhodli sme sa, že sa stĺpce hľadania stredu blesku (stĺpec `milisecond`) a jeho spodnej frekvencie (stĺpec `f_min<2kHz`) pokúsime urobiť na základe farebných pixelov a ich pomerov. Preto sme aj tento spôsob nazvali segmentácia, keďže sme pracovali s jednotlivými segmentmi obrázkov.

Pri práci na tomto oddiele sme v prvom rade potrebovali ovládať základy práce s obrázkami. Na prácu v ňom sme používali obrázok prekonvertovaný na pole. V jednoduchosti sme pre každý pixel disponovali troma hodnotami reprezentujúcimi kanály troch základných farieb – čiže červenej (angl. *red*), zelenej (angl. *green*) a modrej (angl. *blue*), RGB. Vzhľadom k našim potrebám sme pracovali iba s červenou farbou, čiže kanálom R. Na základe toho sme zvyšné dva kanály odstránili.

- `milisecond`

  V tomto stĺpci sa nachádzal údaj o presnej milisekunde v ktorej sa úder blesku uskutočnil. Tento údaj reprezentoval pomyselný stred *eventu*, resp. miesto najväčšej intenzity blesku. Keďže v predikčnom boxe nebol vždy umiestnený daný blesk presne v strede a zároveň pri *tweekoch* bol väčšinou blesk umiestnený na ľavej strane boxu bolo potrebné pristúpiť na prácu s pixelmi. Ako už bolo spomenuté vyššie, na tento stĺpec nám postačovala práca s červeným kanálom. Pre každý označený *event* bol vyseparovaný obrázok a s ním sme následne pracovali. Plán bol vytvoriť skript počítajúci súčet červených bodov po celej výške obrázka, teda po každom stĺpci. Pre lepšiu predstavu je možné jeden zo stĺpcov vidieť na Obrázku 3 – 14. Následne daný program porovnával jednotlivé stĺpce medzi sebou a stĺpec s najvyššou hodnotou červenej farby bol označený za jeho stred. Po tomto kroku sme dostali pozíciu blesku v pixeloch.

  Keďže ale šírka celého spektrogramu bola 1421 pixelov, na vyrátanie jeho hodnoty v milisekundách sme ešte potrebovali hodnotu pixelov vydeliť podľa Rovnice 3.3. A teda za hodnotu *pixel of mid* sme dosadili hodnotu vyrátaného stredu blesku v pixeloch, hodnota 1000 zastupovala počet milisekúnd v jednej

Obrázok 3 – 14 Orientačný náhľad jedného radu pixelov použitého pri vyhodnotení pre stĺpec `milisecond`. Stĺpec pixelov s najvyššou hodnotou červenej farby je zvýraznený čiernou čiarou.

sekunde a hodnota *width of spectrogram* znamenala celkovú šírku spektrogramu, v našom prípade 1421 pixelov.

$$\text{milisecond} = \text{pixel of mid} * \frac{1000}{\text{width of spectrogram}}. \tag{3.3}$$

- `f_min<2kHz`

  Túto hodnotu sme taktiež boli nútení dopočítať podobným spôsobom, ako hodnotu `milisecond`. V tomto prípade sme nepracovali so stĺpcami, teda s vertikálnym usporiadaním pixelov, ale s riadkami. Pri tomto stĺpci tabuľky sme zvolili metódu porovnávania červených pixelov na spodnej časti obrázka nachádzajúcej sa pod 2 kHz a celého obrázka. V prenesení na rozmery – pri každom lokalizovanom *evente* sme zrátali hodnotu všetkých červených kanálov, ktoré sa nachádzali na spodných 20 pixeloch obrázka a porovnali to so súčtom červených kanálov na všetkých pixeloch vyrezaného *eventu*. Tieto dve hodnoty sme dali do pomeru. Hľadaním optimálneho prahu sme sa dostali k hodnote 9,5 %. Ak teda červené kanály spodných 20 pixelov tvorili menej ako 9,5 % červených kanálov na celom obrázku, bolo vo väčšine prípadov jasné, že sa nejedná o *event* s emisiou siahajúcou pod 2 kHz a výstupná hodnota zapísaná do

tabuľky bola 0. V opačnom prípade sa o takýto *event* jednalo a výstupná hodnota bola 1. Hranicu je možné vidieť vyznačenú čiernou horizontálnou čiarou na Obrázku 3 – 15.

Obrázok 3 – 15 Náhľad hranice, na ktorej sme vyhodnocovali či *event* siaha alebo nesiaha pod 2 kHz.

V tomto bode boli všetky dáta pripravené a ostávalo ich už iba spojiť a vygenerovať výstupnú tabuľku.

3.6 Vyhodnotenie

3.6.1 Vyhodnotenie detekcií

V tejto podkapitole sme sa venovali procesu nájdenia najlepšieho *confidence thresholdu*. Tento parameter bol smerodajný z titulu, že menil počet detegovaných *eventov* pri detekciách. Na základe neho sme boli schopní porovnávať presnosť detekcií ako vizuálne, tak aj podľa metrík spomenutých nižšie.

Nakoľko sme v predchádzajúcej podkapitole zvolili ako optimálnu hodnotu *confidence thresholdu* číslo menšie ako 0,50 mohol nastať stav, kedy model označil jeden *event* ako sférik aj *tweek* zároveň. Na základe vizuálnej kontroly sme rozhodli, že vytvoríme skript na vyhodnotenie dát takým spôsobom, že všetky duplicitné predikcie budú označené ako *tweek*. Vo väčšine prípadov (113 zo 136) mali tieto udalosti

aj väčšiu hodnotu pravdepodobnosti predikcie, že sa jedná o *tweek*. Nižšie uvedené kvantitatívne vyhodnotenie nezahŕňa duplicitné označenia. Na vyhodnocovanie modelu sme použili metriky kontingenčnú tabuľku (angl. *confusion matrix*), presnosť (angl. *precision*), návratnosť (angl. *recall*) a F1 skóre (angl. *F1 score*). Kontingenčná tabuľka 3 – 1 popisuje označenia jednotlivých tried, ktoré boli získané na základe výstupnej tabuľky, konkrétne hodnôt `date` a `milisecond`. Jednotlivé hodnoty kontingenčnej tabuľky sme získali na základe porovnávania hodnôt týchto stĺpcov v tabuľke predikcií a tabuľke anotácií. Obišli sme tak vyhodnocovanie IoU predikcií a anotácií, pretože najsilnejšiu emisiu budú mať vždy v rovnakom bode. Tento bod predstavuje hodnotu `milisecond` v tabuľkách.

		Skutočné		
		Sférik	**Tweek**	
Predikované	**Sférik**	TP	FP	Background FP
	Tweek	FN	TN	Background FP
		Background FN	Background FN	

Tabuľka 3 – 1 Kontingenčná tabuľka pre jednotlivé triedy.

Takto získaná kontingenčná tabuľka je používaná na vyhodnocovanie klasifikačného modelu. Je tvorená spravidla dvoma typmi hodnôt, v našom prípade sú to skutočné hodnoty (označované tiež ako *True*) a predikcie (označované tiež ako *Predicted*). Skutočné hodnoty v našom prípade zastupujú reálne hodnoty dostupné z anotácií. Predikcie zastupujú predikované hodnoty ako výstup z klasifikátora. Obe množiny hodnôt pracujú so všetkými klasifikovanými triedami, v našom prípade sférik a *tweek*. Na to, aby kontingenčná tabuľka pokryla všetky existujúce výstupy, pracuje s ďalšími dvoma hodnotami. Pri predikciách sa jedná o falošne negatívne hodnoty (angl. *Background False Negative*) zastupujúce *eventy*, ktoré boli v anotáciách, no model ich nenašiel. Pri skutočných hodnotách sa jednalo o falošne pozitívne *eventy* (angl. *Background False Positive*), ktoré zastupovali množinu *eventov* náj-

dených modelom, no chýbajúcich na anotáciách. Kontingenčná tabuľka obsahuje nasledujúce hodnoty:

- *True Positive* (skr. TP) – táto hodnota v našej práci reprezentovala počet správne označených sférikov,
- *True Negative* (skr. TN) – táto hodnota v našej práci reprezentovala počet správne označených *tweekov*,
- *False Positive* (skr. FP) – táto hodnota poukazovala na počet *tweekov*, ktoré boli klasifikátorom označené ako sférik,
- *False Negative* (skr. FN) – v tejto hodnote sa nachádzal údaj o sférikoch, ktoré boli klasifikátorom označené ako *tweek*,
- *Background False Positive* – v týchto hodnotách sa nachádzala množina *eventov* pre danú triedu, ktoré boli nájdené modelom, no anotátori ich pri anotácii neoznačili,
- *Background False Negative* – v týchto hodnotách sa nachádzala množina *eventov* pre danú triedu, ktoré síce boli označené v anotáciách, no model ich nenašiel.

Použitím týchto hodnôt sme v našej práci vyhodnocovali nasledujúce metriky:

- Metrika presnosť vyhodnocuje pomer správne predikovaných hodnôt a všetkých predikcií danej triedy (alebo celého modelu). Vďaka tomu je možné povedať, že metrika poukazuje na precíznosť modelu. Matematické vyjadrenie metriky je možné vidieť vo Vzťahu 3.4, kde hodnota skutočne pozitívne zastupuje správne predikcie (*event* bol anotovaný a predikovaný rovnako) a menovateľ zastupuje súčet všetkých predikcií modelu danej triedy.

$$\text{Presnosť} = \frac{\text{skutočne pozitívne}}{\text{všetky predikcie}}. \tag{3.4}$$

- Metrika návratnosť vyhodnocuje pomer správne predikovaných hodnôt a všetkých označených vstupov (v našom prípade anotácií). Vo Vzťahu 3.5 je matematické vyjadrenie metriky, kde hodnota skutočne pozitívne zastupuje správne predikcie menovateľ zastupuje všetky anotácie danej triedy. V anglickom jazyku sa na menovateľ používa pojem *all ground truths*.

$$\text{Návratnosť} = \frac{\text{skutočne pozitívne}}{\text{všetky anotácie}}. \tag{3.5}$$

- F1 skóre je metrika vyhodnocujúca vzťah medzi presnosťou a návratnosťou. Matematicky je táto metrika popísaná na vzťahu nižšie. Zaoberá sa vyváženosťou hodnôt dvoch vyššie spomenutých metrík a poukazuje na celkovú správnosť modelu.

$$\text{F1 skóre} = 2 * \frac{\text{presnosť} * \text{návratnosť}}{\text{presnosť} + \text{návratnosť}}. \tag{3.6}$$

Ako bolo spomenuté v predchádzajúcej kapitole, po porovnaní jednotlivých predikcií sme skonštatovali, že najlepšia hodnota *confidence threshold* na predikciu našom datasete je približne 0,30. Ďalej sme na základe metrík presnosť, návratnosť, F1 skóre a kontingenčnej tabuľky dolaďovali jeho presnú hodnotu a hľadali optimálny pomer všetkých *eventov*. Porovnávali sme metriky na rozsahu 0,25 – 0,35. Na základe týchto metrík a zároveň fyzickej kontroly obrázkov sme nakoniec vyhodnotili, že z celého rozsahu sa najlepšie javili výsledky pri hodnote *conf* 0,30.

Trieda	**Obrázky**	**Eventy**	**Presnosť**	**Návratnosť**	**F1 skóre**
Spolu	219	2135	0,72	0,66	0,69
Sférik	219	1890	0,74	0,68	0,71
Tweek	219	245	0,58	0,56	0,57

Tabuľka 3 – 2 Tabuľka zobrazujúca metriky modelu pri hodnotách IoU 0,45 a *conf* 0,30. Hodnoty pre triedu Spolu predstavujú vážený priemer.

Na Tabuľke 3 – 2 je možné vidieť všetky metriky pre predikciu s hodnotami IoU 0,45 a *conf* 0,30. Stĺpec `Obrázky` zobrazuje počet obrázkov pre jednotlivé triedy, na ktorých bola predikcia vykonaná. Na stĺpci `Eventy` je zobrazený počet *eventov* množiny z anotácií, na ktorých bola vykonaná predikcia a na základe ktorých boli vypočítavané metriky. Stĺpce `Presnosť`, `Návratnosť` a `F1 skóre` poukazujú na rovnomenné metriky. Na základe tabuľky môžeme vyhodnotiť, že náš model predikuje lepšie sfériky, ako *tweeky*. Môže to byť zapríčinené aj tým, že trénovacia množina obsahovala niekoľkonásobne viac sférikov ako *tweekov*.

		Skutočné		
		Sférik	**Tweek**	**Background FP**
Predikované	**Sférik**	1278	71	379
	Tweek	62	138	39
	Background FN	550	36	0

Tabuľka 3 – 3 Kontingenčná tabuľka pre jednotlivé triedy.

Tabuľka 3 – 3 zobrazuje kontingenčnú tabuľku obsahujúcu počty predikcií jednotlivých tried spolu so zaradením. Pri vertikálnom pohľade na tabuľku (po stĺpcoch) je možné vidieť skutočné hodnoty, ktoré boli na vstup modelu dané ako anotácie. Pri pohľade na tabuľku po riadkoch je zasa možné odčítať počet predikcií pre každú triedu. Stĺpce `FN` a `FP` poukazujú na hodnoty, ktoré vôbec neboli nájdené, no nachádzali sa na anotáciách, resp. boli nájdené navyše. Nakoľko je toto vyhodnotenie vytvorené na základe údajov stredu *eventu* získaných segmentáciou je možné, že niektoré stredy *eventov* boli posunuté o jeden pixel a tým pádom nebola nájdená zhoda. V tomto prípade mohol byť *event* označený ako *Background False Negative*, no algoritmus ho mohol vyhodnotiť ako *Background False Positive*, teda navyše. Väčšina *eventov* spadajúcich do tejto chyby by sa takto mohla navzájom anulovať. Predpokladáme však, že v budúcej práci po normalizácii údajov z analyzátora ELMAVAN-G na rovnaký rozsah sa minimalizuje šum a segmentačné techniky budú

vyhodnocovať stred s väčšou istotou.

3.6.2 Vyhodnotenie segmentácie

Aby sme vedeli vyhodnotiť správnosť segmentácie pod 2 kHz a nad 2 kHz, potrebovali sme ručne anotovať testovaciu množinu dát. Výsledky základných metrík segmentácie v porovnaní s anotáciami sú uvedené v Tabuľke 3 – 4. Tabuľka 3 – 5 predstavuje kontingenčnú tabuľku, kde môžeme vidieť, že 79 udalostí bolo chybne klasifikovaných ako *eventy*, ktorých emisia nesiahala pod 2 kHz, pričom v skutočnosti tieto *eventy* pod zvolenú hranicu siahali, t.j. falošne pozitívne a 42 falošne negatívnych udalostí.

Trieda	**Presnosť**	**Návratnosť**	**F1 skóre**	**Počet**
Spolu	0,94	0,94	0,94	1967
nad 2 kHz	0,93	0,96	0,94	1053
pod 2 kHz	0,95	0,91	0,93	914

Tabuľka 3 – 4 Vyhodnotenie metrík segmentácie, či emisia *eventu* siaha pod 2 kHz, alebo nie. Hodnoty pre triedu Spolu predstavujú vážený priemer.

		Skutočné	
		nad 2 kHz	pod 2 kHz
Predikované	nad 2 kHz	1011	79
	pod 2 kHz	42	835

Tabuľka 3 – 5 Kontingenčná tabuľka pre vyhodnotenie segmentácií.

Po množstve vytvorených experimentov sa nám týmto nastavením podarilo splniť merateľné kritériá, ktoré boli zvolené na začiatku výskumu.

3.7 Nasadenie

Tým, že bol program vyvíjaný na základe konkrétnej spolupráce a zadaného dopytu, bude dostupný a používaný pracovníkmi Ústavu fyziky atmosféry AV ČR. Aktuálne prístupný dataset nie je jednotný – frekvenčná škála intenzity udalostí na spektrogramoch je v súčasnosti relatívna – líši sa od jednotlivých spektrogramov. Po doručení datasetu obsahujúceho spektrogramy s intenzitou vyjadrenou farbami v absolútnych hodnotách bude tento program nasadený na všetky takéto dáta a jeho výstup použitý na ďalšiu analýzu a štatistické spracovanie. Ako veľmi dôležitá sa javí nielen samotná detekcia a klasifikácia do dvoch tried, ale aj určenie, či sférik siaha pod 2 kHz, alebo nie. Tento jav určuje, akým spôsobom sa šírila elektromagnetická vlna od blesku k detektoru. Signál s frekvenciou pod 2 kHz sa môže šíriť iba v špecifickom móde, pretože jeho vlnová dĺžka je približne 150 km, zatiaľ čo výška medzi zemou a D vrstvou ionosféry je približne 90 km. V jednoduchosti sa dá povedať, že určením, či sa detegovaná emisia sfériku nachádza aj pod 2 kHz vieme určiť základné rozdelenie charakteristík elektromagnetickej vlny šíriacej sa od blesku ku detektoru. Ako bolo v práci preukázané, jedná sa o veľký objem dát, ktorý nie je fyzicky spracovateľný a teda možné očakávať, že daný program bude adekvátne využitý. Program bude spustený jednorázovo na všetky existujúce dáta. Spustenie programu na analýzu dát v reálnom čase zatiaľ v pláne nie je, no v budúcnosti to nie je vylúčené.

4 Záver

Neurónové siete sú všade okolo nás a čím ďalej tým viac sa stávajú súčasťou našich životov. Vďaka požiadavke od AV ČR sme dostali impulz na využitie neurónových sietí aj tam, kde ešte v súčasnosti nemajú majoritné zastúpenie pri riešení analytických úloh. Základný cieľ práce bol natrénovať neurónovú sieť schopnú detegovať a klasifikovať *eventy* na spektrogramoch a na základe jej výstupov vyrobiť výstupnú tabuľku obsahujúcu dáta o každom konkrétnom *evente*. Vďaka obsiahlemu zdroju dát a *know-how*, ktoré konzultanti práce mali, bolo možné plnohodnotne implementovať toto riešenie a z veľkej miery splniť požiadavku na nástroj pre dodatočnú analýzu dát.

Priamym pokračovaním tejto práce je samotné nasadenie na celú databázu dát, ktorá bude zdrojom pre ďalší výskum. Zdokonaľovanie predikcií by bolo možné overiť natrénovaním nového modelu, ktorý bol vývojármi YOLOv5 vydaný v čase písania práce. Tento model obsahuje váhy natrénované na obrázkoch s väčším rozlíšením (1280 pixelov), čo by malo zaručiť vyššiu presnosť pri predikciách. Anotačný projekt zároveň prináša informáciu o počte *tweekov*, ktorého účelom bude v budúcej práci vytvoriť ďalšiu neurónovú sieť, ktorá bude predikovať počet disperzií jednotlivých *tweekov*. Zároveň sa v budúcej práci je možné zamerať na nekontrolované učenie a porovnať výsledky s výsledkami predloženej bakalárskej práce.

Práca priniesla autorovi dôležité skúsenosti s prácou s neurónovými sieťami, extrakciou textov, no aj pohľad na obrázky z tzv. druhej strany, kedy autor pracoval s hodnotami farebného kanála RGB. Vzhľadom na časovú obtiažnosť práce a vysoké množstvo úprav dát a práce s nimi sú tieto skúsenosti upevnené a autor ich je schopný využiť a prezentovať sa nimi neskôr v ďalšej vedeckej práci.

Literatúra

Bottou, L. (2012). Stochastic gradient descent tricks, *Neural networks: Tricks of the trade*, Springer, pp. 421–436.

Brabham, D. C. (2008). Crowdsourcing as a model for problem solving: An introduction and cases, *Convergence* **14**(1): 75–90.

Brownlee, J. (2019). A gentle introduction to the rectified linear unit (relu), *Machine learning mastery* **6**.

Chollet, F. et al. (2018). *Deep learning with Python*, Vol. 361, Manning New York.

Duchi, J., Hazan, E. and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization., *Journal of machine learning research* **12**(7).

Fanioudakis, L. and Potamitis, I. (2017). Deep networks tag the location of bird vocalisations on audio spectrograms, *arXiv preprint arXiv:1711.04347* .

Ghosh, S., Das, N., Das, I. and Maulik, U. (2019). Understanding deep learning techniques for image segmentation, *ACM Computing Surveys (CSUR)* **52**(4): 1–35.

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*, MIT Press. `http://www.deeplearningbook.org`.

Jaeger, P. F., Kohl, S. A., Bickelhaupt, S., Isensee, F., Kuder, T. A., Schlemmer, H.-P. and Maier-Hein, K. H. (2020). Retina u-net: Embarrassingly simple exploitation of segmentation supervision for medical object detection, *Machine Learning for Health Workshop*, PMLR, pp. 171–183.

Jocher, G., Stoken, A., Borovec, J., NanoCode012, Chaurasia, A., TaoXie, Changyu, L., V, A., Laughing, tkianai, yxNONG, Hogan, A., lorenzomammana, AlexWang1900, Hajek, J., Diaconu, L., Marc, Kwon, Y., oleg, wanghaoyang0106,

Defretin, Y., Lohia, A., ml5ah, Milanko, B., Fineran, B., Khromov, D., Yiwei, D., Doug, Durgesh and Ingham, F. (2021). ultralytics/yolov5: v5.0 - YOLOv5-P6 1280 models, AWS, Supervise.ly and YouTube integrations.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization, *arXiv preprint arXiv:1412.6980* .

Konan, O. J., Mishra, A. K. and Lotz, S. (2020). Machine learning techniques to detect and characterise whistler radio waves, *arXiv preprint arXiv:2002.01244* .

Lin, T.-Y., Goyal, P., Girshick, R., He, K. and Dollár, P. (2017). Focal loss for dense object detection, *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988.

Liu, S., Qi, L., Qin, H., Shi, J. and Jia, J. (2018). Path aggregation network for instance segmentation, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8759–8768.

McGonagle, J., Shaikouski, G., Williams, C. et al. (2018). Backpropagation. brilliant. org.

Neubeck, A. and Van Gool, L. (2006). Efficient non-maximum suppression, *18th International Conference on Pattern Recognition (ICPR'06)*, Vol. 3, IEEE, pp. 850–855.

Noriega, L. (2005). Multilayer perceptron tutorial, *School of Computing. Staffordshire University* .

O'Shea, T. J., Roy, T. and Erpek, T. (2017). Spectral detection and localization of radio events with learned convolutional neural features, *2017 25th European Signal Processing Conference (EUSIPCO)*, IEEE, pp. 331–335.

O'Shea, T., Roy, T. and Clancy, T. C. (2017). Learning robust general radio signal detection using computer vision methods, *2017 51st Asilomar Conference on Signals, Systems, and Computers*, IEEE, pp. 829–832.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. (2016). You only look once: Unified, real-time object detection, *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Ronneberger, O., Fischer, P. and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation, *International Conference on Medical image computing and computer-assisted intervention*, Springer, pp. 234–241.

Saha, S. (2018). A comprehensive guide to convolutional neural networks—the eli5 way, *Towards Data Science* **15**.

Santolík, O. and Kolmašová, I. (2017). Unusual electromagnetic signatures of european north atlantic winter thunderstorms, *Scientific reports* **7**(1): 1–9.

Sharma, S. (2017). Activation functions in neural networks, *towards data science* **6**.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude, *COURSERA: Neural networks for machine learning* **4**(2): 26–31.

Wang, C.-Y., Mark Liao, H.-Y., Wu, Y.-H., Chen, P.-Y., Hsieh, J.-W. and Yeh, I.-H. (2020). Cspnet: A new backbone that can enhance learning capability of cnn, *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 390–391.

Yap, M. H., Hachiuma, R., Alavi, A., Brungel, R., Goyal, M., Zhu, H., Cassidy, B., Ruckert, J., Olshansky, M., Huang, X. et al. (2020). Deep learning in diabetic foot ulcers detection: A comprehensive evaluation, *arXiv preprint arXiv:2010.03341* .

Zoznam príloh

Príloha A CD médium – záverečná práca v elektronickej podobe, príručky v elektronickej podobe a zdrojový kód.

Príloha B Používateľská príručka

Príloha C Systémová príručka